Система контроля Насосно-Аккумуляторной Станции подачи воды.

Оглавление

1 Введение

1.1 Область применения

До сих пор в Украине существуют множество домохозяйств, в которых о стабильной подаче питьевой воды можно только мечтать. Выкручиваться из сложившейся ситуации как всегда домовладельцам приходиться самостоятельно. Поэтому во многих частных домах устанавливаются насосно-аккумуляторные станции (НАС) воды.

Построение их может отличаться. Краткое описание некоторых из них:

- насос подает воду в домашний водопровод из скважины. Кроме него установлены механический фильтр воды, реле давления, аккумулятор воды, запорно-распределительная гидроаппаратура;
- вода из центрального трубопровода поступает в большой накопительный резервуар, а уже из него воду в домашний водопровод подает НАС, дополнительное оборудование то же;
- развитие и комбинирование указанных способов.

1.2 Существующие недостатки

Указанное оборудование решает вопросы подачи воды, но в то же время появляются дополнительные факторы отказа системы, которые могут привести к аварийным ситуациям, таким как:

- протечка воды после насоса (обычно оборудование установлено вдали от жилых помещений, поэтому об аварии можно узнать далеко не сразу);
- отсутствие воды в источнике (скважина или бак) при этом насос продолжает работать всухую, перегревается и выходит из строя. Датчиков наличия воды в штатном оборудовании не предусмотрено.
- падение давления воздуха в аккумуляторе, что приводит к более длительной работе насоса (вплоть до постоянной работы). Отключение реле давления настраивается на сумму давлений воздуха в аккумуляторе и верхнего давления воды в системе. Учитывая, что воздух из бака убегает всегда, это довольно частая ситуация;
- изменение температуры окружающей среды также приводит к изменению времени срабатывания по уставкам реле;
- прокол мембраны аккумулятора так же приводит к подобной ситуации;
- засорение гидравлических каналов реле давления;
- искрение, перегрев и как следствие залипание контактов реле давления, управляющим включением насоса. Что может привести к длительной работе насоса, перегреву реле и пожару, бесконтрольном повышении давления в системе, порывам и протечкам.

После опробования некоторых аварийных ситуаций сформировалась необходимость разработать и изготовить систему контроля НАС подачи воды.

2 Анализ требований

2.1 Технологические требования

Перед контроллером ставятся следующие технологические задачи:

- контроль наличия напряжения AC220В на электродвигателе насоса;
- контроль потребляемого тока электродвигателем насоса;
- измерение давления воды или воздуха в аккумуляторе;
- учет и контроль времени работы насоса за одно включение;
- учет и контроль времени работы насоса за сутки;
- контроль температуры окружающей среды;
- контроль протечки воды;
- принудительное включение/выключение насоса;
- принудительная блокировка включения насоса;
- сигнализация и предупреждение аварийных ситуаций;
- локальная индикация основных технологических параметрах системы;
- доступ к параметрам системы через веб-сервер.

2.2 Аппаратные требования

Для решения указанных задач лучше всего подойдет контроллер из бюджетных серий для широкого применения, для которого существует хороший выбор готовых модулей.

Выбранный контроллер должен решить указанные выше технологические задачи, к которым добавляются множество технических задач, таких как:

- преобразование уровней сигналов для приведения к уровням контроллера (AC220В – DC5В);
- преобразование переменного сигнала значения тока к пропорциональному среднеквадратичному сигналу приведённый к уровню аналогового входа контроллера;
- исполнительное оборудование для цепей электродвигателя насоса;
- дистанционная передача и прием информации;
- часы реального времени;
- подключение к локальной информационной TCP/IP сети;
- память хранения кода веб-сервера;
- согласование различных уровней сигналов контроллеров и модулей;
- питание контроллеров и модулей;

2.3 Требование к Программному обеспечению

Ввиду уникальности разработки программное обеспечение для работы контроллеров также необходимо разработать. Для его разработки необходимы прикладное ПО, среда разработки и готовые библиотеки для подключаемого оборудования.

3 Структура системы контроля НАС

В разработке структуры всей системы контроля НАС в первую очередь надо исходить из существующего расположения оборудования, мест контроля, точек подключения к информационной сети.

В моем случае оборудование НАС расположено в защищенном от внешних воздействий цокольном этаже. В нем есть питание AC220В и нет информационной сети (Wi-Fi).

Место где желательно контролировать НАС конечно находится в жилом помещении. В котором есть питание и информационная сеть (Wi-Fi).

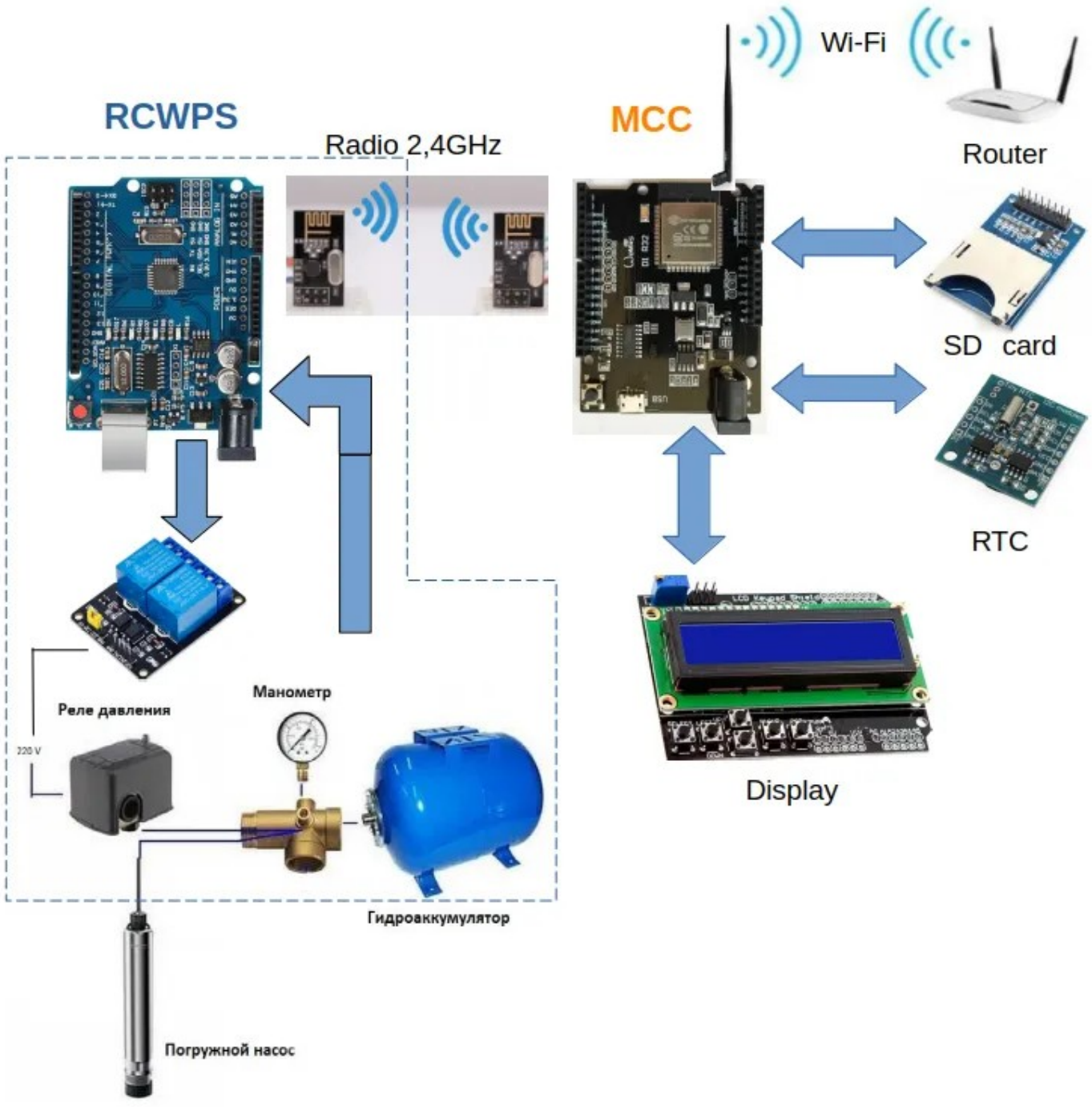

Рисунок 1 — Структурная схема системы контроля НАС подачи воды

Исходя из этого Система контроля НАС состоит из двух частей (контроллеров). Один расположен в помещении НАС, назовем его Remote Controller Water Pump Station (RCWPS), другой в жилом помещении, назовем его Master Central Controller (MCC). Задачи и подключаемые модули разделяются между ними.

Выше приведена структурная схема системы контроля НАС подачи воды. Информация от оборудования НАС поступает в контроллер RCWPS, который выполняет:

- базовую обработку сигналов;
  - проверку на аварийные ситуации;
  - обработку местных команд управления;
  - управление промежуточными реле в цепи электродвигателя;
  - упаковку данных;
  - передачу / прием на / от базового контроллера MCC.

Связь между контроллерами выполнена по беспроводному каналу диапазона 2,4 ГГц.

Контроллер MCC выполняет:

- прием / передачу информации от / на контроллер RCWPS;
  - распаковку данных;
  - управление счетчиками времени работы (таймеры);
  - обмен информацией с часами реального времени RTC;
  - обмен информацией с энергонезависимой памятью;
  - определяет расширенные аварийные ситуации;
  - обрабатывает команды пользователя;
  - управляет сигнализацией;
  - выводит параметры на дисплей;
  - выводит параметры на последовательный порт;
  - обмен информацией с картой памяти SD;
  - организовывает работу веб-сервера.

Подключение к беспроводной сети Wi-Fi будет выполнять контроллер MCC.

4 Аппаратная часть

4.1 Оборудование НАС

Оборудование НАС подачи воды можно разделить на гидравлическое (Насос, фильтры, трубопроводы, реле давления) и электрическое оборудование (электродвигатель, автоматический выключатель, реле давления, коммутационные цепи).

Настоящей разработкой вносим незначительные изменения в электрическую схему питания электродвигателя насоса и подключаем измерительные цепи.

Даже не устанавливая систему контроля НАС в существующую схему между реле давления и электродвигателем следует добавить промежуточный контактор К1. Он разгружает силовые цепи реле давления и избавляет от неисправностей, связанных с подгоранием и залипанием контактов реле.

Установка контактора К1 позволяет осуществлять дистанционное управление НАС. Для чего в схему устанавливаются реле KV1, KV2 – Дистанционный Пуск / Стоп насоса и реле KV3 – блокировка включения насоса.

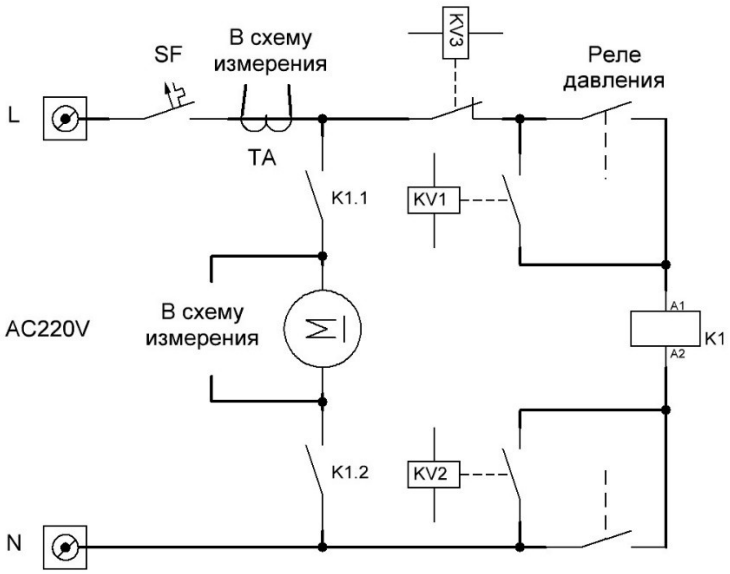

Рисунок 2 — Электрическая схема подключения НАС подачи воды

4.2 Контроллер RCWPS

4.2.1 Плата контроллера

В качестве удаленного контроллера выбрал один из клонов Arduino UNO на микроконтроллере ATmega328. Он соответствует необходимым требованиям за небольшие деньги, а именно:

- серийно выпускаемый контроллер из бюджетной линейки;
- наличие серийно выпускаемых модулей для него;
- наличие бесплатной среды разработки Arduino IDE и библиотек;
- тактовая частота 16 МГц;
- объем Flash памяти 32Кб, RAM 2Кб, EEPROM 1Кб;

- наличие 14 дискретных входов / выходов;
- наличие 6 аналоговых входов;
- имеет USB-B с преобразователем интерфейсов USB-UART для программирования;
- питание 7-12В через стандартный разъем.

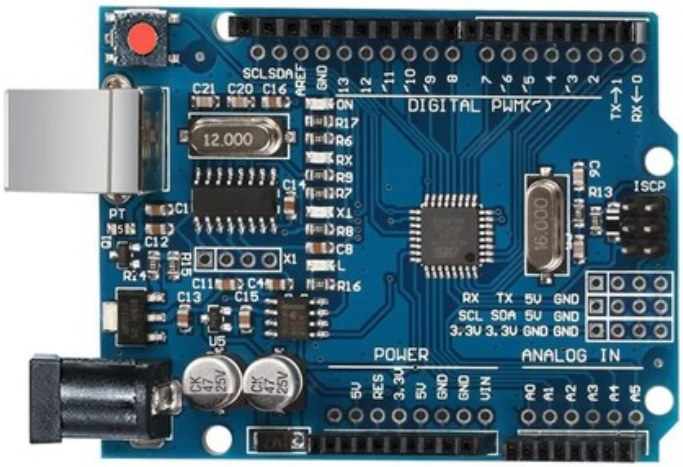

Рисунок 3 – Плата контроллера Arduino UNO

4.2.2 Модуль реле

Модуль 2 реле 5V с опторазвязкой предназначен для возможности подключения низковольтных управляющих сигналов контроллера к цепям питания AC220В.

Модуль состоит из двух реле, каждое из которых может коммутировать ток до 10А при напряжении 220В переменного тока. Обмотки реле питаются от 5В постоянного тока.

Для подключения в схему понадобилось добавить еще одно реле параллельно существующему. То есть на плате установлены KV1 и KV3, а параллельно контактам катушки KV1 добавил KV2.

Рисунок 4 – Плата модуля 2 реле 5V с опторазвязкой

4.2.3 Модуль трансивера

Беспроводной модуль 2.4GHz NRF24L01+. Выполнен на однокристальном ВЧ-трансивере безлицензионного ISM-диапазона 2,4ГГц…2,5ГГц, состоящий из интегрального синтезатора частоты, усилителя мощности, кварцевого генератора, демодулятора, модулятора и модуля аппаратного протокола. Данный модуль полностью берет на себя функции приема / передачи с анализом доставки пакета.

Основные характеристики: частотный диапазон 2,4ГГц…2,5ГГц, 80 каналов, скорость передачи данных до 2Мбит/с, выходная мощность 0дБм, напряжение питания 1,9В…3,6В.

Рисунок 5 – Плата беспроводного модуля 2.4GHz NRF24L01+

4.2.4 Датчик температуры и влажности

Рисунок 6 – Датчик температуры и влажности DHT11

4.2.5 Модуль определения переменного напряжения

Теперь о нестандартных модулях. Определение работы электродвигателя насоса будем определять по подаче на него напряжения AC220B. Для этого проще всего самостоятельно спаять необходимый **модуль сопряжения AC220B**.

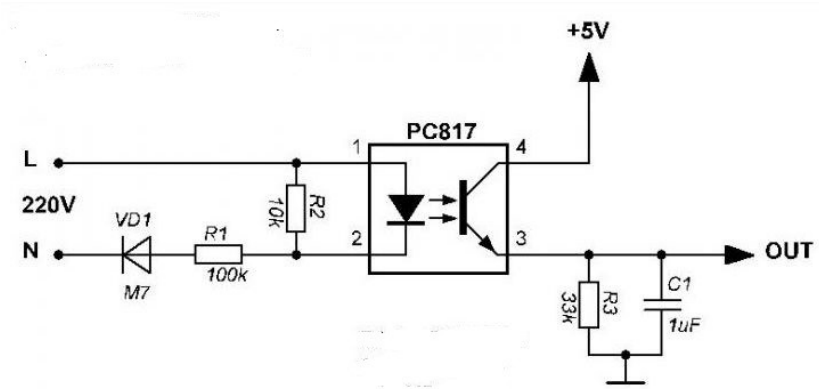

Рисунок 7 – Схема сопряжения цепи AC220B со входом контроллера

4.2.6 Модуль измерения переменного тока

То же самое касается **модуля измерения переменного тока** электродвигателя насоса. В стандартной схемотехнике это достаточно насыщенная схема преобразования на операционных усилителях. Для народного применения задача интегрирования переменного входного сигнала выполняется программной обработкой. В итоге схема упрощается просто до ввода переменного

сигнала в допустимом диапазоне напряжений с искусственно установленным нулем.

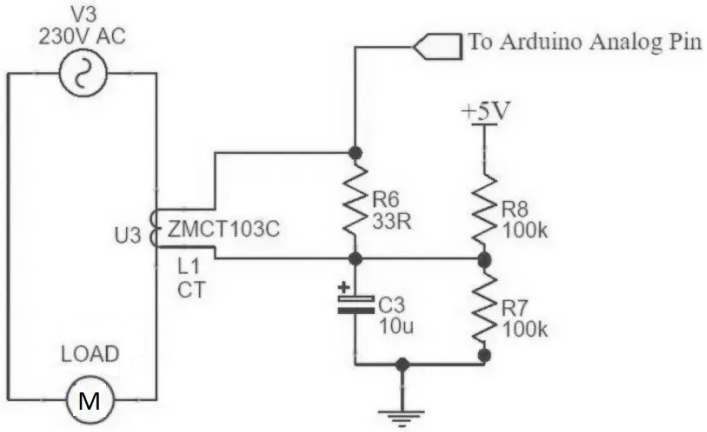

Рисунок 8 – Схема сопряжения трансформатора тока со входом контроллера

Обе схемы сопряжения разместились на одной плате (нижняя). Сверху **плата** с сенсорными кнопками **управления** работой насоса (START, STOP, BLOCK WORK) и индикацией предупреждения об аварийных ситуациях.

Рисунок 9 – Плата местного управления и плата сопряжения цепей AC220B

4.2.7 Модуль питания

Питание контроллера и модулей удобно выполнить через **плату питания** на макетную плату HW-131. При входном напряжении: 6.5В-12В (через стандартный разъем DC5521) получаем выходные напряжения 5В и 3,3В. Максимальный выходной ток 700мА (суммарный ток по обоим напряжениям).

Рисунок 10 – Плата питания HW-131

4.2.8 Общий вид

После размещения контроллера и модулей на едином монтажном шасси получилось изделие – контроллер RCWPS. Следует сразу оговориться что при

изготовлении внешний вид не принимался во внимание, использовались надежные компоненты из наличия или по минимальной стоимости.

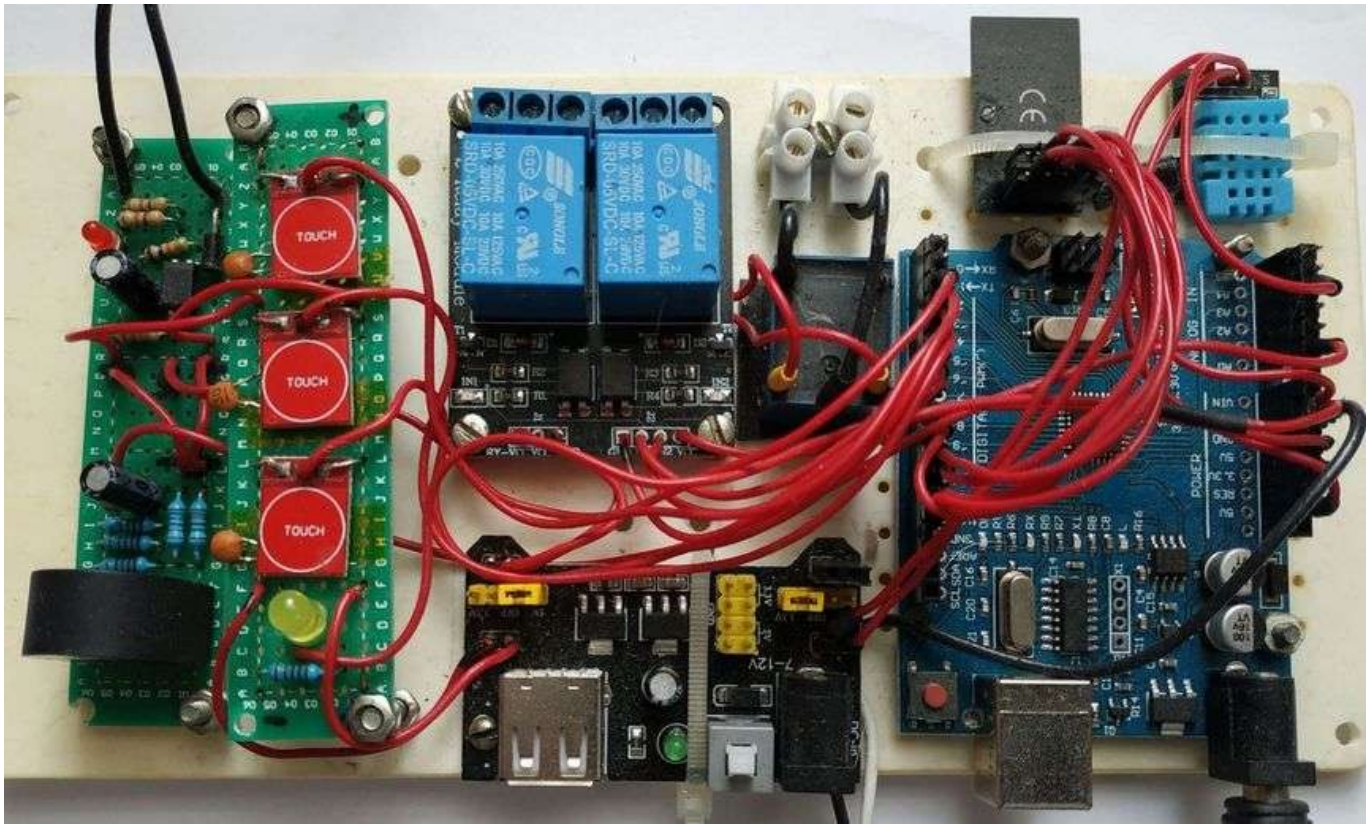

Рисунок 11 – Контроллер RCWPS

4.3 Контроллер MCC

4.3.1 Плата контроллера

На центральный контроллер ложится гораздо больше вычислительных и коммуникационных задач. Для этого подойдет плата контроллера WEMOS D1 R32 на микроконтроллере ESP32 фирмы Espressif. Его основные характеристики:

- серийно выпускаемый контроллер из бюджетной линейки;
- возможность подключать серийные модули для Arduino;
- возможность использовать Arduino IDE;
- наличие библиотек;
- тактовая частота 240МГц;
- объем Flash памяти 4Мб, RAM 512Кб;
- 20 дискретных входов / выходов;
- 6 аналоговых входов;
- коммуникационные интерфейсы SPI, I2C, I2S, IR, UART, PWM;
- Wi-Fi 802.11 b/g/n и Bluetooth 4.2 LE;
- встроенная антенна Wi-Fi;
- имеет micro-USB с преобразователем интерфейсов USB-UART для программирования;
- питание контроллера и цепей ввода – вывода 3,3В;
- питание 7-12В через стандартный разъем.

Рисунок 12 – Плата контроллера Wemos D1 R32

4.3.2 Модуль трансивера

Такой же Беспроводной модуль 2.4GHz NRF24L01+ для связи с удаленным контроллером.

Рисунок 13 – Плата беспроводного модуля 2.4GHz NRF24L01+

4.3.3 Карта памяти

Примененный контроллер обладает встроенным интерфейсом SD-MMC, что позволяет оставить больше выводов ввода – вывода для периферии. Поскольку питание контроллера 3,3В то карту памяти можно подключать напрямую, без преобразования уровней. Использован стандартный адаптер SD – micro SD, к которому подпаяны провода. В него уже установлена карта памяти micro SD.

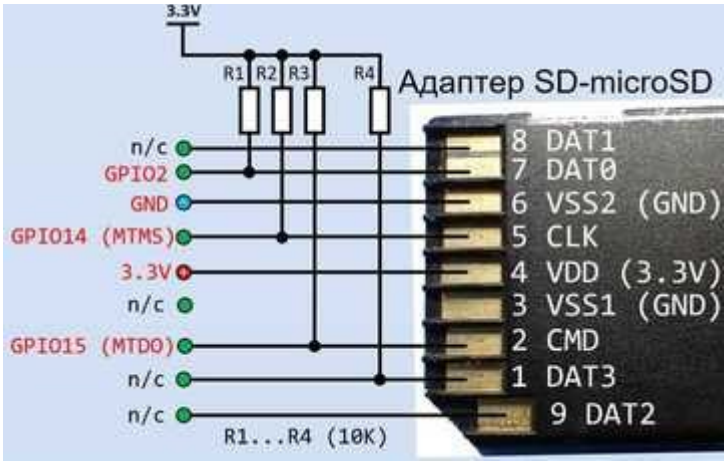

Рисунок 14 – Схема подключения памяти SD-MMC

4.3.4 Модуль индикации и управления

Плату индикации и управления собрал из двухстрочного дисплея LCD1602, модуля конвертера i2c to LCD1602, кнопок и светодиодов. Модуль преобразовывает параллельный интерфейс дисплея в последовательный I2C. Кнопки имеют тоже назначение как на удаленном контроллере.

START – пуск насоса на ограниченное время (1 мин), ограничение сделано с целью предотвратить аварийные ситуации при длительной работе насоса, поскольку включение происходит в обход реле давления;

STOP – отключение насоса, не влияет если насос включен от реле давления;

BLOCK WORK – блокировка включения насоса. Если насос включен останавливает его. Работает как триггер, для включения необходимо повторная команда.

Поскольку насос работает автоматически, а перечисленные выше команды нужны в особых случаях то кнопки утоплены в корпус, для нажатия используется толкатель.

MENU – есть только на центральном контроллере. Предназначена для переключения экранов на дисплее. Так же включает подсветку дисплея на ограниченное время (3 мин). Для облегчения доступа выполнена как сенсорная кнопка, электродом прикосновения выступает правый нижний винт крепления.

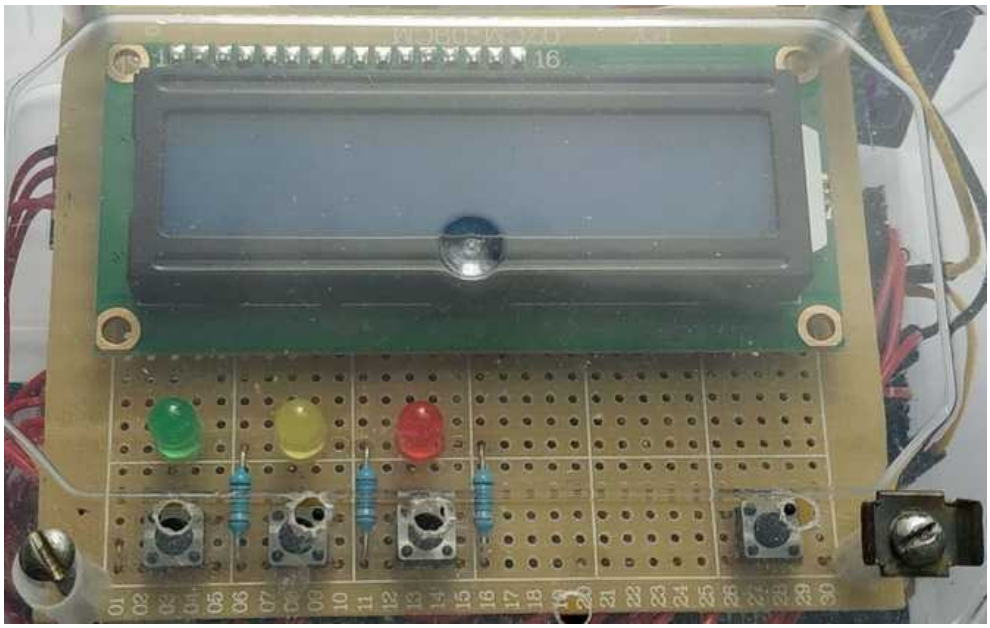

Рисунок 15 – Плата индикации и управления

Поскольку количество выводимой информации больше того, что может вместить дисплей выполнено разделение на экраны выводимой информации по группам:

Экран 1. Счетчики и таймеры;

Экран 2. Значения измеряемых параметров;

Экран 3. Сообщения об ошибках.

4.3.5 Модуль RTC

Часы реального времени RTC DS1307 I2C собраны на основании двух микросхем: DS1307 (часы реального времени) и AT24C32 (EEPROM на 32K bit память).

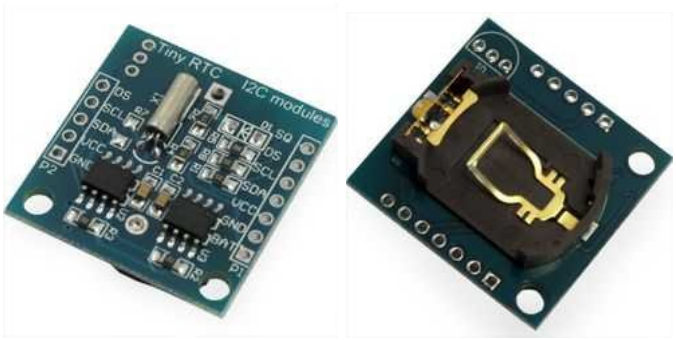

Рисунок 16 – Часы реального времени RTC DS1307

Модуль конвертера дисплея и часы реального времени подключаются по последовательному интерфейсу I2C, всего по двум проводам.

4.3.6 Преобразователь уровней

Обоим устройствам для нормальной работы требуется напряжение питания 5В. Поэтому для связи с контроллером (уровни 3,3В) требуется двунаправленный преобразователь уровней.

Рисунок 17 – Модуль преобразователь уровней 5В – 3,3В 4х канальный

4.3.7 Модуль питания

Питание дополнительных плат и модулей организовано на двух отдельных **платах стабилизаторов** серии 1117 5В и 3,3В. Стабилизатор контроллера собственный.

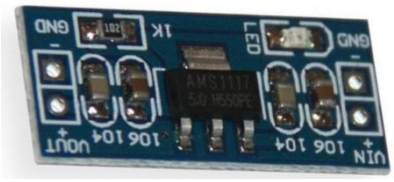

Рисунок 18 – Плата стабилизатора серии 1117

4.3.8 Общий вид

После сборки плат и модулей в корпус получил изделие контроллер МСС. Опять стоит напомнить, что при изготовлении внешний вид не принимался во внимание, использовались надежные компоненты из наличия или по минимальной стоимости.

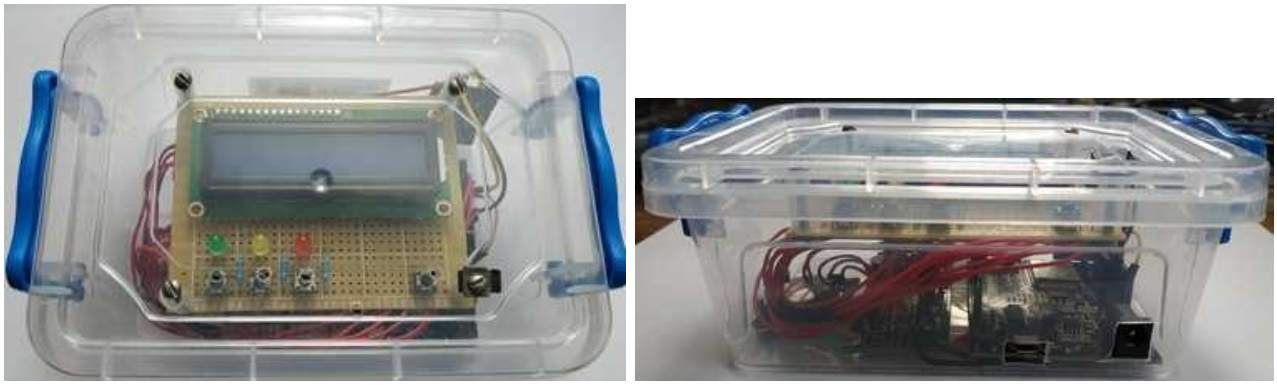

Рисунок 19 – Общий вид контроллера МСС

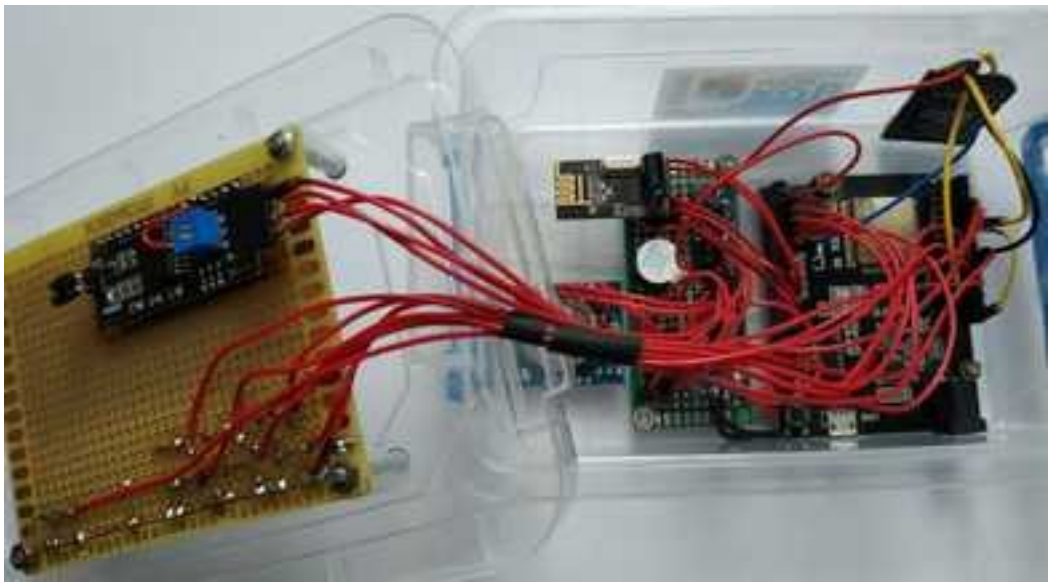

Рисунок 20 – Контроллер МСС с открытой крышкой

5 Программное обеспечение

5.1 Архитектура программного обеспечения

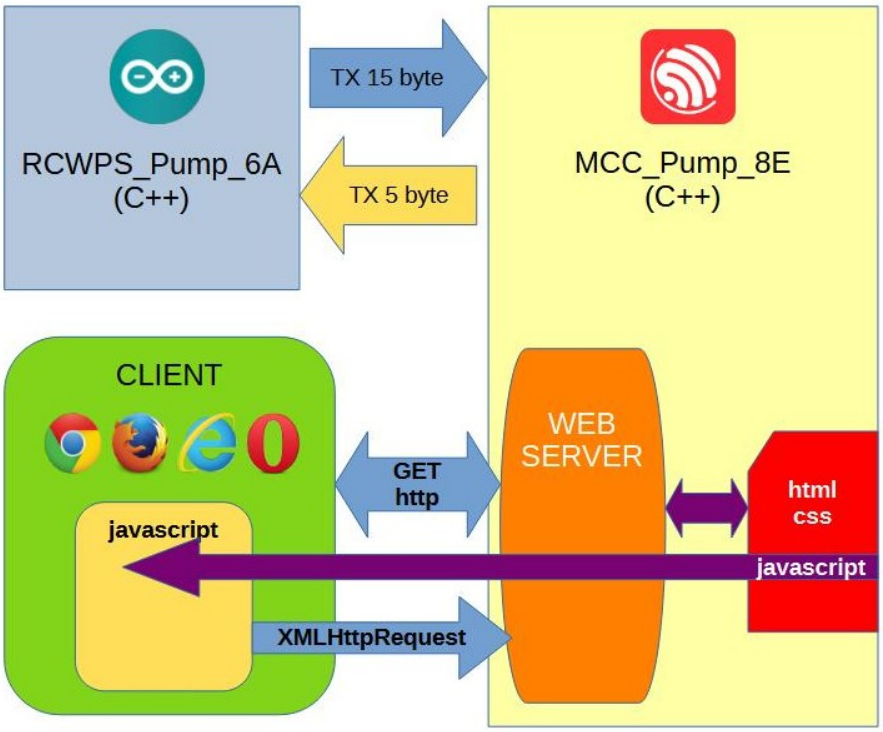

Рисунок 21 – Структурная схема информационного обмена

Рассмотрим основные принципы информационного обмена. На рисунке показаны возможные участники прохождения информации. Контроллеры RCWPS, MCC, веб-сервер на его основе, клиент.

Как уже упоминалось передача / прием информации между контроллерами выполняется пакетными посылками по радиоканалу. Назначение посылки будет рассмотрено ниже.

На основании полученной информации контроллерами выполняется контроль и управление оборудованием НАС, сигнализация состояний, вывод на дисплей параметров.

В контроллере MCC кроме управляющей программы запущен веб-сервер, подключенный к локальной сети через встроенный сетевой интерфейс. При подключении клиента по IP адресу сервера, порт 80 ему передается считанная с SD карты html/css страница и скрипт исполняемой программы. В браузере клиента будет отображаться полученная страница, а получение обновляемых параметров (каждую секунду) обеспечит скачанный скрипт по протоколу XMLHttpRequest. Отправка команд от клиента выполняется в методе GET.

Итого система контроля НАС имеет четыре объекта с программным обеспечением:

- управляемое прикладное программное обеспечение контроллера RCWPS, C++;
- управляемое прикладное программное обеспечение контроллера MCC, C++;

- веб сервер;
- сайт для клиента (html/css) со скриптом клиента (JavaScript).

5.2 Пакетная передача информации

Прием и передача информации между контроллерами выполняется пакетами по радиоканалу. При передаче от контроллера RCWPS длина пакета составляет 15 байт. Для передачи от контроллера MCC зарезервировано 5 байт. Имена переменных оставлены собственные, назначение указано в листинге программы.

Таблица 1 - Назначение сигналов в пакете передачи от контроллера MCC

BYTE	Bit / Name Variables							
	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
0	alarmMCC	commandOnPump	commandOffPump	commandBlock	-	-	-	-
1	Reserved							
2	Reserved							
3	Reserved							
4	Reserved							

Таблица 2 – Значения ошибок в переменной codErrMCC

bit	Описание ошибки codErrMCC
0	one switching time is longer than 10 minutes
1	working time per day longer than 60 minutes
2	the consumed current is above the nominal value
3	the consumed current is below the nominal value
4	the temperature is above 30C
5	Reserved
6	Reserved
7	Reserved

Таблица 3 - Назначение сигналов в пакете передачи от контроллера RCWPS

BYTE	Bit / Name Variables							
	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
0	voltagePumpPin	buttonOnPumpPin	buttonOffPumpPin	buttonBlockPin	-	-	-	-
1	alarmRCWPS	onPumpPin	blockPumpPin	-	-	-	-	-
2	high byte (currentRMS * 100)							
3	low byte (currentRMS * 100)							
4	Reserved							
5	Reserved							
6	high byte (Temperature * 10)							
7	low byte (Temperature * 10)							
8	byte Humidity							
9	Error code RCWPS (codErrRCWPS[8])							
10	Rest time to OFF Pump - cycle							
11	Reserved							
12	Reserved							
13	Reserved							
14	Reserved							

Таблица 4 – Значения ошибок в переменной codErrRCWPS

bit	Описание ошибки codErrRCWPS
0	If output onPump is ON (inverse) and no voltage
1	If output blockPump is ON (inverse) and voltage on
2	Reserved
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved

5.3 Контроллер RCWPS

ПО написано на C++, среда разработки Arduino IDE https://github.com/Ochilnik/WPS/blob/master/RCWPS_Pump_6A.ino.

Программное обеспечение для контроллеров строится по принципу постоянной работы по циклу. Для этого в языке C++ Arduino IDE существует специальная функция loop(). Все что находится в ней постоянно будет выполняться по кругу. Причем выполнение алгоритма не должно приводить к тупиковым ветвям или значительно изменять время выполнения машинного цикла.

Выполнению программы по циклу предшествует выполнение функции setup(), которая выполняется один раз после включения или рестарта контроллера.

Исходя из этого функциональный код разделен на части.

5.3.1 Подключенные библиотеки

SPI.h - библиотека для работы с шиной SPI (подключен модуль nRF2401);

nRF24L01.h, RF24.h - библиотеки для работы с модулем nRF2401;

Adafruit_Sensor.h - библиотека для работы с датчиками;

DHT.h, DHT_U.h - библиотеки для работы с датчиком температуры и влажности DHT11;

EmonLib.h - библиотека Emon Library для измерения переменного тока.

5.3.2 Переменные и константы

```
#define DHTPIN 17                    // Digital pin connected to the DHT sensor
#define DHTTYPE    DHT11             // DHT 11

RF24 radio(9, 10);                   // порты D9, D10: CE, CSN  For Arduino UNO
DHT_Unified dht(DHTPIN, DHTTYPE);
String bits = "";
EnergyMonitor emon1;                 // Create an instance

const uint64_t addr_tx = 0xAABBCCDD11LL;  // адрес коннектора на передачу
const uint64_t addr_rx = 0xAABBCCDD22LL;  // адрес коннектора на прием
const uint8_t num_ch = 119;          // номер канала

bool commandAlarm;                   // command from MCC - Alarm
bool commandOnPump;                  // command from MCC - Pump ON
bool commandOffPump;                 // command from MCC - Pump OFF
bool commandBlock;                   // command from MCC - Block Pump
bool stateAlarm, alarmMCC, alarmRCWPS;    // Current States signals from RCWPS
bool codErrRCWPS[8] = {0};           // Array of errors codes from RCWPS

uint8_t data_tx[15] = {0};           // Array data to transmit
uint8_t data_rx[5] = {0};            // Array data to recieve
uint8_t data_tx_previos[15] = {0};   // Array data that transmit last time
uint8_t cycleTimer = 14;             // Количество циклов таймера до нужного времени 60s / 4,19424s = 14,3
uint8_t cycle = cycleTimer;          // Текущее Количество циклов таймера
//uint8_t codErrMCC;                 // code of errors from MCC

float currentCalibration = 27.0;
float currentRMS;

unsigned long measurement_timestamp = 0;
unsigned long interval = 4000ul;  // Интервал измерений DTH в миллисекундах
```

```
//***** PINOUT *****
uint8_t voltagePumpPin = 2; // Voltage is supplied to the motor pump
uint8_t buttonOnPumpPin = 4; // button pump on is pressed
uint8_t buttonOffPumpPin = 5; // button pump off is pressed
uint8_t buttonBlockPin = 3; // triger blocking pump is on
uint8_t blockPumpPin = 6; // command blocking pump
uint8_t onPumpPin = 7; // command pump on
uint8_t alarmPin = 8; // signal attention
uint8_t currentPumpPin = 14; // current is consumed motor pump

void setup(){
// ***** Settings INPUT/OUTPUT
  pinMode(voltagePumpPin, INPUT); // Voltage is supplied to the pump motor
  pinMode(buttonOnPumpPin, INPUT); // button pump on is pressed
  pinMode(buttonOffPumpPin, INPUT); // button pump off is pressed
  pinMode(buttonBlockPin, INPUT); // triger blocking pump is on
  pinMode(blockPumpPin, OUTPUT); // command pump off
  digitalWrite(blockPumpPin, !LOW);
  pinMode(onPumpPin, OUTPUT); // command pump on
  digitalWrite(onPumpPin, !LOW);
  pinMode(alarmPin, OUTPUT); // signal attention
}
```

Листинг 1 – Модуль Переменные и константы

5.3.3 Пакетный прием / передача

Это основная функция контроллера. Как видно в листинге рабочему процессу предшествует объемная часть предустановок. Закомментированные параметры оставлены для удобства наладки.

```
void setup(){
// ***** Settings Radio Interface
  Serial.println("Reciver/Transmitter ON");
  radio.begin(); // инициализация радиомодуля
  delay(2000);
  //radio.setPayloadSize(7);        // Установить статичный размер блока данных пользователя в байтах (0-32)
  radio.setDataRate(RF24_250KBPS);  // Установить скорость передачи RF24_1MBPS, RF24_2MBPS, RF24_250KBPS
  switch(radio.getDataRate()){                    //  Проверка установленной скорости radio.getDataRate()
    case RF24_1MBPS   : Serial.println("setDataRate  1 MBPS"); break; // Если полученное значение RF24_1MBPS
    case RF24_2MBPS   : Serial.println("setDataRate  2 MBPS"); break; // Если полученное значение RF24_2MBPS
    case RF24_250KBPS : Serial.println("setDataRate  250 KBPS"); break;
  }
  radio.setCRCLength(RF24_CRC_8);   // Установить размер CRC 8 bit или 16 bit
  radio.setChannel(num_ch);         // установка номера канала
  radio.setAutoAck(true);           // подтверждение приема
  radio.setPALevel(RF24_PA_MAX);    // мощность RF24_PA_MIN, RF24_PA_LOW, RF24_PA_HIGH  RF24_PA_MAX

  //-----Reciever part--
  //radio.enableDynamicPayloads();  //включить поддержку динамического размера полезной нагрузки на приемнике
  radio.openReadingPipe(1, addr_rx); // открыть коннектор на прием
  //closeReadingPipe();             // Закрыть трубу открытую ранее для прослушивания (приёма данных).
  //radio.enableAckPayload();       // разрешить отсылку данных в ответ на входящий сигнал
  radio.startListening();           // перевод модуль в режим работы приемника

  //-----Transmitter part--
  radio.setRetries(5, 15);          // время ожидания = (число+1) * 250 мкс и количество попыток отправки данных от 0 до 15
  radio.openWritingPipe(addr_tx);   // открыть коннект на отправку
  //radio.stopListening();          // перевод модуля в режим работы передатчика
  //radio.powerUp();                // режим полной нагрузки
  //radio.powerDown();              // режим пониженного потребления
}
void loop(){
//****** Reciever part
  if(radio.available()){                    // Если в буфере имеются принятые данные, то ...
    radio.read(&data_rx, sizeof(data_rx));     // Читаем данные из буфера в массив myData
```

```
  //RX BYTE 0
  alarmMCC = (data_rx[0] >> 0) & 0x01; // Bit 0. Remote Command to Alarm
  commandOnPump = (data_rx[0] >> 1) & 0x01; // Bit 1. Remote Command to On Pump
  commandOffPump = (data_rx[0] >> 2) & 0x01; // Bit 2. Remote Command to Off Pump
  commandBlock = (data_rx[0] >> 3) & 0x01; // Bit 3. Remote Command to Block work Pump
  //RX BYTE 1
  //
 }

//****** Transmitter part
 // Prepare data
 //TX BYTE 0
 data_tx[0] =  (digitalRead(buttonBlockPin) << 3) | (digitalRead(buttonOffPumpPin) << 2) |
            (digitalRead(buttonOnPumpPin) << 1) | digitalRead(voltagePumpPin);

 //TX BYTE 1
 data_tx[1] =  (!digitalRead(blockPumpPin) << 2) |
            (!digitalRead(onPumpPin) << 1) | alarmRCWPS;

 //TX BYTES 2 - 3. Motors current
 currentRMS = emon1.calcIrms(1480);  // Calculate Irms only
 int packI = (int)round(currentRMS * 100);
 // Storage Pump Current (2 байта)
 data_tx[2] = (packI >> 8) & 0xFF; // Старший байт
 data_tx[3] = packI & 0xFF;        // Младший байт

 //TX BYTES 4 - 5. Reserved for 15Ai - Waters pressure

 //TX BYTES 6 - 8
 if (millis() - measurement_timestamp >= interval) {
  sensors_event_t event;
  // Get temperature event and print its value.
  dht.temperature().getEvent(&event);
  if (!isnan(event.temperature)) {
   int packTemp = (int)round(event.temperature * 10);
   // Сохраняем температуру (2 байта)
   data_tx[6] = (packTemp >> 8) & 0xFF; // Старший байт
   data_tx[7] = packTemp & 0xFF;        // Младший байт
  }
  // Get humidity event and print its value.
  dht.humidity().getEvent(&event);
  if (!isnan(event.relative_humidity)) {
   data_tx[8] = (int)round(event.relative_humidity);
  }
  measurement_timestamp = millis(); // Обновление метки времени
 }

 //TX BYTE 9. Error code RCWPS
 data_tx[9] = 0;
 for (int i = 0; i < 8; i++) {
  if (codErrRCWPS[i]) {
   data_tx[9] |= (1 << i); // Установить бит на соответствующей позиции
  }
 }

 //TX BYTE 10. Rest time to OFF Pump
 data_tx[10] = cycle;

 // Transmit data if not equal previous
 if(!dataAreEqual(data_tx, data_tx_previos, sizeof(data_tx))) {
  radio.stopListening();                  // Выключаем приемник, завершаем прослушивание открытых труб.
  radio.write(&data_tx, sizeof(data_tx));      // Отправляем данные из массива myData
  toSerialTX();
  radio.startListening();                 // Включаем приемник, начинаем прослушивать открытые коннекты.
 }

 // Copy current station to previous station
 for (uint8_t i = 0; i < sizeof(data_tx); i++) {
  data_tx_previos[i] = data_tx[i];
 }
```

```
}
```

Листинг 2 - Модуль пакетного приема / передачи

В рабочем цикле сперва осуществляется прием и раскодирование принятой информации по глобальным переменным. Затем выполняется подготовка пакета для передачи. Библиотечная функция передачи, передает до 32 байт в пакете (в проекте используется 15 байт).

При этом для эффективного использования доступного адресного пространства пакета переменные типа BOOL упаковываются в байты по группам. Переменные типа INT наоборот разделяются на раздельные байты – старший и младший.

Хотя пакет данных подготавливается каждый рабочий цикл, передается он только если отличается от пакета, сформированного в предыдущем цикле. Для этого после сравнения пакетов и передачи выполняется копирование подготовленного пакета во временный массив.

Данный подход позволяет значительно разгрузить зашумленность радиоэфира и энергопотребление контроллера без снижения надежности связи.

5.3.4 Таймер пуска

Отсчет времени в контроллерах требуется выполнять, не влияя на выполнение основного цикла программы. Поэтому для отсчета времени после принудительного пуска насоса использован аппаратный таймер 1 микроконтроллера и стандартные библиотеки управления таймерами с помощью бит слова управления.

Использован сигнал переполнения таймера при максимальной выдержке:

- делитель тактовой частоты 1024, 16 МГц / 1024 = 15625 кГц;
- время одного такта 1 / 15625 кГц = 0,000064 сек;
- время работы таймера до переполнения 65535 * 0,064 = 4,19424 сек.

Даже при максимальных уставках таймера время выдержки недостаточно. Поэтому введена дополнительная переменная (cycleTimer = 14) количества циклов для которой организован программный счет, что позволяет достичь желательной выдержки около 1 мин.

Особенностью подпрограмм аппаратных прерываний это необходимость быстрого их выполнения, а значит минимальное количество кода в них. Поэтому при прерывании устанавливается флаг или изменяется счетчик, а остальной алгоритм выполняется в основной программе.

Таймер управляется включением / отключением его в отдельных функциях управления пуска / стопа насоса.

```
void setup(){
// ***** Settings Timer1
  noInterrupts();               // отключаем все прерывания
  TCCR1A = 0;                     //  управляющие регистры таймера/счетчика
  TCCR1B = 0;
  TCNT1 = 0;                     // регистр для установки заранее загружаемого значения
```

```
  TIMSK1 |= (1 << TOIE1);              // enable timer overflow interrupt ISR
  interrupts();                        // разрешаем все прерывания
}

void startTimer1(){
  TCCR1B |= (1 << CS10)|(1 << CS12);    // Start timer 1 with prescaler 1024
}

void stopTimer1(){
  TCCR1B &= ~(1 << CS10 | 1 << CS11 | 1 << CS12);    // Stop Timer1
}

ISR(TIMER1_OVF_vect) {  // процедура обработки прерывания переполнения счетчика
  if(cycle>0){
     cycle--;
  } else {
     stopPump();
  }
```

Листинг 3 - Модуль таймера

5.3.5 Управление

Теперь на управляющую часть программы остается только определить форсированную команду от кнопок или от центрального контроллера и вызвать соответствующую функцию.

```
void startPump(){
  digitalWrite(onPumpPin, !HIGH);       // Start Pump (inverse)
  startTimer1();                        // start Timer1 on 63s = cycleTimer(14) * 4,19424s
}

void stopPump(){
  digitalWrite(onPumpPin, !LOW);        // Stop Pump (inverse)
  cycle = cycleTimer;                   // Reset cycles
  TCNT1 = 0;                            // Reset Timer1
  stopTimer1();
}

void blockPump(){
  stopPump();
  digitalWrite(blockPumpPin, !HIGH);    // Disconnect Pump from power
  }

void unblockPump(){
  digitalWrite(blockPumpPin, !LOW);     // Connect Pump to power
}

void loop(){

//********** Control part - Commands and Buttons
  if(digitalRead(onPumpPin) && digitalRead(blockPumpPin) && (commandOnPump || digitalRead(buttonOnPumpPin))) {
    startPump();
  }
  if(!digitalRead(onPumpPin) && digitalRead(blockPumpPin) && (commandOffPump || digitalRead(buttonOffPumpPin))) {
    stopPump();
  }
  if(digitalRead(blockPumpPin) && (commandBlock || digitalRead(buttonBlockPin))) {
    blockPump();
  }
  if(!digitalRead(blockPumpPin) && (!commandBlock && !digitalRead(buttonBlockPin))) {
    unblockPump();
  }
}
```

Листинг 4 - Модуль форсированного управления

5.3.6 Обработка аварийных ситуаций

Функции обработки аварийных ситуаций поделены между двумя контроллерами. Для возможности автономной работы в контроллере RCWPS определяются только базовые ошибки, которые можно обработать без получения дополнительной информации от центрального контроллера. Контроллер MCC уже учитывает ошибки для определения которых требуется использовать изменяемые или вычисляемые параметры.

Перечень ошибок указан в главе Пакетная передача информации. Как видно по оставленному резерву запас для экспериментов еще есть.

```
bool isError() {
  for(uint8_t i=0; i < 8; i++){
    if(codErrRCWPS[i] == true){
      alarmRCWPS = true;
      return true;
    }
  }
  alarmRCWPS = false;
  return false;
}

void loop(){
//****** Alarms
  for (int i = 0; i < 8; i++) {                    // reset array of alarms
  codErrRCWPS[i] = false;
  }
  if(!digitalRead(onPumpPin) && !digitalRead(voltagePumpPin)){     // If output onPump is ON (inverse) and no voltage
    codErrRCWPS[0] = true;                         // set error code
  } else {
    codErrRCWPS[0] = false;                        // reset error code
  }
  if(!digitalRead(blockPumpPin) && digitalRead(voltagePumpPin)){  // If output blockPump is ON (inverse) and voltage on
    codErrRCWPS[1] = true;                         // set error code
  } else {
    codErrRCWPS[1] = false;                        // set error code
  }
  // Control Alarm LED
  if(isError() || alarmMCC){
    digitalWrite(alarmPin, HIGH);
  } else {
    digitalWrite(alarmPin, LOW);
  }
}
```

Листинг 5 - Модуль определения аварийных ситуаций

5.4 Контроллер MCC

Контроллер ESP32 имеет разные инструментарии разработки. Использовалась среда разработки Arduino IDE с подключенными модулями ф.Espressif. ПО для написано на C++ https://github.com/Ochilnik/WPS/blob/master/MCC_Pump_8E.ino.

Использовались те же принципы как в ранее описанном контроллере. Поскольку ESP32 имеет другую архитектуру некоторые функции имеют особенности. Далее приведено краткое описание кода исходя из функционального назначения по частям.

5.4.1 Подключенные библиотеки

SPI.h - библиотека для работы с шиной SPI (подключен модуль nRF2401);

nRF24L01.h, RF24.h - библиотеки для работы с модулем nRF2401;

Wire.h - библиотека для работы с шиной I2C (подключены модули LCD, RTC);

LiquidCrystal_I2C.h - библиотека для работы с LCD дисплеем по шине I2C;

driver/timer.h - библиотека для работы с таймером ESP-IDF;

uRTCLib.h - библиотека для работы с DS1307 по шине I2C;

SD_MMC.h, FS.h – библиотеки для работы с SD-MMC;

WiFi.h - библиотека для работы с WiFi;

5.4.2 Переменные, константы и объекты

```
#define TIMER_GROUP TIMER_GROUP_0     // Группа таймеров
#define TIMER_IDX   TIMER_0           // Индекс таймера
#define TIMER_DIVIDER 40000           // Предделитель (APB тактирование / 40000 = 2 kГц)
#define TIMER_ALARM_VALUE 240000      // Срабатывание через 30 секунд (в 0,5 милисекундах)
#define NUM_AVG 10                    // Количество циклов для вычисления среднего значения тока
#define REQ_BUF_SZ   120              // size of buffer used to capture HTTP requests
#define ADRDAY 1                      // 1 byte address currentDay in I2C AT24C32 memory (addr: 1 - 56)
#define ADRNUMON 2                    // 1 byte address numOnPump in I2C AT24C32 memory (addr: 1 - 56)
#define ADRCOUNT 3                    // 2 bytes address countOnPump in I2C AT24C32 memory (addr: 1 - 56)
#define ADRCOUNTDAY 5                 // 2 bytes address vountDayPump in I2C AT24C32 memory (addr: 1 - 56)

LiquidCrystal_I2C lcd(0x27,16,2);     // Объявляем  объект lcd, (адрес I2C = 0x27, столбцов = 16, строк = 2)
RF24 radio(13, 5);                    // Объявляем  объект radio. порты D13 - CE, D5 - CSN
String bits = "";                     // Объявляем  строковый объект bits

byte rtcModel = URTCLIB_MODEL_DS1307; // Set RTC model
uRTCLib rtc;                          //  Объявляем  объект rtc - часы

File webFile;                         // the web page file on the SD card
char HTTP_req[REQ_BUF_SZ] = {0};      // buffered HTTP request stored as null terminated string
char req_index = 0;                   // index into HTTP_req buffer

const char *ssid = "SSID";            // name WiFi
const char *password = "password";    // password WiFi
NetworkServer server(80);             // Объявляем сетевой объект server


const uint64_t addr_tx = 0xAABBCCDD22LL;  // адрес коннекта на передачу
const uint64_t addr_rx = 0xAABBCCDD11LL;  // адрес коннекта на прием
const uint8_t num_ch = 119;               // номер канала
const float timeCycle = 4.19424;          // One cycle time is Arduino UNO counter

bool voltageOn;                       // Voltage is ON to pump
bool commandOnPump;                   // command to ON pump from RCWPS (Remote Controller for Water Pump Station)
bool commandOffPump;                  // command to OFF pump from RCWPS (Remote Controller for Water Pump Station)
bool commandBlock;                    // command to Block pump from RCWPS (Remote Controller for Water Pump Station)
bool stateAlarm, alarmMCC, alarmRCWPS, stateOnPump, prevStateOnPump, stateBlock // States signals from RCWPS
bool trigerBlock = 0, prevBlock = 0;  // trigger flags
bool tMenu, prevMenu;                 // touch button Menu, previos state button Menu
bool codErrMCC[8];                    // code of errors from MCC
bool codErrRCWPS[8];                  // code of errors from RCWPS
volatile bool backLight = false;      // Global state LCD light
volatile bool flagLight = false;      // Global flag state LCD light
volatile bool flagCount = false;      // Global flag enable counters from rtc
volatile bool flagPulse = false;      // Global flag is pulse from rtc.sq 1Hz
bool webOn, webOff, webBlock;         // command ON, OFF, Block from web client
bool pumpON, pumpOFF;                 // summ commands from MCC & WebClient
```

```
uint8_t data_tx[5] = {0};               // Array data to transmit
uint8_t data_rx[15] = {0};              // Array data to recieve
uint8_t data_tx_previos[5] = {0};       // Array data that transmit last time
uint8_t numOnPump;                      // number of pump starts
uint8_t currentDay;                     // today
uint8_t numScreen = 1;                  // number current screen on Display
const uint8_t lcdScreens = 3;           // number screens lcd Display
uint16_t year;                          // current year
uint8_t month;                          // current month
uint8_t day;                            // current day
uint8_t hour;                           // current hour
uint8_t minute;                         // current minute
uint8_t dayHour;                        // hours in day counter
uint8_t dayMinute;                      // minutes in day counter
uint8_t daySecond;                      // seconds in day counter
uint8_t tHour;                          // hours in last counter
uint8_t tMinute;                        // minutes in last counter
uint8_t tSecond;                        // seconds in last counter
uint8_t lowByte;                        // Младший байт
uint8_t highByte;                       // Старший байт

uint16_t countOnPump;                   // time last start
uint16_t countDayPump;                  // pump operating time for the whole day
const uint16_t maxOnPump = 600;         // max time pump is working at once
const uint16_t maxDayPump = 3600;       // max time pump is working at day

int timeToOff = 0;                      // Оставшееся время до отключения насоса
int humidity;                           // Humiduty from RCWPS (Remote Controller for Water Pump Station)
int pressure;

float currentRMS;                       // RMS current value
float temperature;                      // Temperatures from RCWPS (Remote Controller for Water Pump Station)
const float maxCurrent = 2.95;          // max RMS current value
const float minCurrent = 2.5;           // min RMS current value
const float maxTemp = 30;               // max temperatures
float avgCurrent;                       // average currentRMS

// Pinout configuration
uint8_t buttonMenu = 4;         // button menu on is pressed
uint8_t buttonOnPumpPin = 27;   // button pump on is pressed
uint8_t buttonOffPumpPin = 32;  // button pump off is pressed
uint8_t buttonBlockPin = 33;    // triger blocking pump is on
uint8_t ledOnPumpPin = 16;      // led pump is on
uint8_t ledAlarmPin = 17;       // led alarm is on
uint8_t ledBlockPin = 25;       // led blocking pump is on
uint8_t buzzerPin = 26;         // buzzer alarm is on
uint8_t rtcSQPin = 35;          // rtc wave 1Hz input

void setup(){
// ******************************** Settings INPUT/OUTPUT ***********************
  //pinMode(buttonMenu, INPUT_PULLUP);      //  button menu
  pinMode(buttonOnPumpPin, INPUT_PULLUP);   // button pump on
  pinMode(buttonOffPumpPin, INPUT_PULLUP);  // button pump off
  pinMode(buttonBlockPin, INPUT_PULLUP);    // button blocking pump
  pinMode(rtcSQPin, INPUT);                 // SQ wave from RTC
  pinMode(ledOnPumpPin, OUTPUT);            // led pump is on
  pinMode(ledAlarmPin, OUTPUT);             // led signal attention
  pinMode(ledBlockPin, OUTPUT);             // led blocking pump is on
  pinMode(buzzerPin, OUTPUT);               // sound signal attention
}
```

Листинг 6 – Модуль Переменные, константы и объекты

5.4.3 Пакетный прием / передача

Уже знакомая функция контроллера. Предустановки практически идентичны предыдущему контроллеру, за исключением адресов коннектов на прием и

передачу поменянных местами. Закомментированные параметры оставлены для удобства наладки.

```
void setup(){
// ************************** Settings Radio Interface ***************************
  Serial.println("Reciver/Transmitter ON");
  radio.begin();                  // инициализация радиомодуля
  delay(2000);
  //radio.setPayloadSize(7);        // Установить статичный размер блока данных пользователя в байтах (0-32)
  radio.setDataRate(RF24_250KBPS);  // скорость передачи данных RF24_1MBPS, RF24_2MBPS или RF24_250KBPS
  switch(radio.getDataRate()){                  //  Проверка установленной скорости radio.getDataRate()
    case RF24_1MBPS   : Serial.println("setDataRate  1 MBPS"); break; // Если полученное значение RF24_1MBPS
    case RF24_2MBPS   : Serial.println("setDataRate  2 MBPS"); break; // Если полученное значение RF24_2MBPS
    case RF24_250KBPS : Serial.println("setDataRate  250 KBPS"); break;
  }
  radio.setCRCLength(RF24_CRC_8);   // Установить размер CRC 8 bit или 16 bit
  radio.setChannel(num_ch);         // установка номера канала
  radio.setAutoAck(true);           // подтверждение приема
  radio.setPALevel(RF24_PA_MAX);    // усилитель RF24_PA_MIN, RF24_PA_LOW, RF24_PA_HIGH или RF24_PA_MAX

  //-----------------------Reciever part-----------------------
  //radio.enableDynamicPayloads();  //включить поддержку динамического размера полезной нагрузки на приемнике
  radio.openReadingPipe(1, addr_rx); // открыть коннектор на прием
  //closeReadingPipe();             // Закрыть коннект открытый ранее для прослушивания (приёма данных).
  //radio.enableAckPayload();       // разрешить отсылку данных в ответ на входящий сигнал
  radio.startListening();           // перевод модуль в режим работы приемника

  //---------------------Transmitter part--------------------------
  radio.setRetries(5, 15);          // время ожидания = (число+1) * 250 мкс и количество попыток отправки данных от 0 до 15
  radio.openWritingPipe(addr_tx);   // открыть коннект на отправку
  //radio.stopListening();          // перевод модуля в режим работы передатчика
  //radio.powerUp();                // режим полной нагрузки
  //radio.powerDown();              // режим пониженного потребления
}
void loop(){
//********************** Reciever part ****************************************************88
  if(radio.available()){                    // Если в буфере имеются принятые данные, то ...
    radio.read(&data_rx, sizeof(data_rx));     // Читаем данные из буфера в массив myData

    //RX BYTE 0
    voltageOn = (data_rx[0] >> 0) & 0x01;       // Bit 0. Remote signal that voltage is on to Pump
    commandOnPump = (data_rx[0] >> 1) & 0x01;   // Bit 1. Remote Command to On Pump
    commandOffPump = (data_rx[0] >> 2) & 0x01;  // Bit 2. Remote Command to Off Pump
    commandBlock = (data_rx[0] >> 3) & 0x01;    // Bit 3. Remote Command to Block work Pump

    //RX BYTE 1
    alarmRCWPS = (data_rx[1] >> 0) & 0x01;      // Bit 0. State Alarm
    stateOnPump = (data_rx[1] >> 1) & 0x01;     // Bit 1. State On Relay to ON Pump
    stateBlock = (data_rx[1] >> 2) & 0x01;      // Bit 2. State On Relay to Block work Pump

    //RX BYTES 2 - 3. Motors current
    uint16_t current_raw = (data_rx[2] << 8) | data_rx[3]; // Объединяем два байта
    currentRMS = current_raw / 100.0;             // Преобразуем обратно в float

    //RX BYTES 4 - 5. Reserved for 15Ai - Waters pressure

    //RX BYTES 6 - 8. Temperature and humidity
    uint16_t temperature_raw = (data_rx[6] << 8) | data_rx[7]; // Объединяем два байта
    temperature = temperature_raw / 10.0;             // Преобразуем обратно в float
    humidity = data_rx[8];

    //RX BYTE 9. Array of errors codes from RCWPS
    for (int i = 0; i < 8; i++) {
      codErrRCWPS[i] = data_rx[9] & (1 << i); // Проверить бит на позиции i и сохранить в массив
    }

    //RX BYTE 10. Rest time to off pump
    timeToOff = static_cast<int>(timeCycle * data_rx[10]);
  }
```

```
//******************** Transmitter part ************************
  // Prepare data
  //TX BYTE 0
  data_tx[0] =  (trigerBlock << 3) | (pumpOFF << 2) | (pumpON << 1) | alarmMCC;
  //TX BYTE 1
  //TX BYTE 2
  //TX BYTE 3
  //TX BYTE 4

  // Transmit data if not equal previous
  if(!dataAreEqual(data_tx, data_tx_previos, sizeof(data_tx))) {
    radio.stopListening();                    // Выключаем приемник, завершаем прослушивание открытых труб.
    radio.write(&data_tx, sizeof(data_tx));      // Отправляем данные из массива myData.
    // SerialPrint tx data. Генерация строки битов
    Serial.println("      TX");
    bits = "";
    for (int i = 7; i >= 0; i--) {
       bits += String((data_tx[0] >> i) & 1);  // Сдвиг вправо и побитовая операция AND
    }
    Serial.println("      Inputs = " + bits);
    Serial.println();
    radio.startListening();                   // Включаем приемник
  }

  // Copy current station to previous station
  for(uint8_t i = 0; i < sizeof(data_tx); i++) {
    data_tx_previos[i] = data_tx[i];
  }
}
```

Листинг 7 - Модуль пакетного приема / передачи

В рабочем цикле сперва осуществляется прием и раскодирование принятой информации по переменным. Затем выполняется подготовка пакета для передачи. Библиотечная функция передачи, передает до 32 байт в пакете (в проекте используется 5 байт).

При этом для эффективного использования доступного адресного пространства пакета переменные типа BOOL упаковываются в байты по группам.

Пакет данных подготавливается каждый рабочий цикл, передается только если он отличается от пакета, сформированного в предыдущем цикле. Для этого после сравнения пакетов и передачи выполняется копирование подготовленного пакета во временный массив.

Данный подход позволяет значительно разгрузить зашумленность радиоэфира и энергопотребление контроллера без снижения надежности связи.

5.4.4 Дисплей

Как уже упоминалось параметры на дисплей выводятся несколькими экранами.

Экран 1. Счетчики и таймеры;

Экран 2. Значения измеряемых параметров;

Экран 3. Сообщения об ошибках.

Переключение между ними осуществляется с помощью кнопки MENU. На экране 3 сообщения об ошибках обновляются по кругу.

```
void setup(){
```

```
// ************************ Settings LCD on I2C ************************
  lcd.init();                      //  Инициируем работу с LCD дисплеем
}

void loop(){
  // switch to around LCD Screens after MENU button pressing
  //if(!digitalRead(buttonMenu) && backLight && prevMenu){    // ver for button Menu
  if(tMenu && backLight && prevMenu){                         // ver for touch Menu
    numScreen++;
    if(numScreen > lcdScreens){
      numScreen = 1;
    }
  }
  //prevMenu = digitalRead(buttonMenu);
  prevMenu = tMenu;
toDisplay();
}

// function out information on LCD display. use global variables
void toDisplay() {
    // Previous state of variables
    static bool pv;         // voltageOn
    static bool par;        // alarmRCWPS
    static bool pam;        // alarmMCC
    static bool psop;       // stateOnPump
    static bool psb;        // stateBlock
    static uint8_t pnd;     // number screen on display
    static uint8_t phour;     // Hour
    static uint8_t pminute;      // Minute
    static uint8_t pnop;    // numOnPump
    static uint8_t ph;      // humidity
    static uint16_t pcop;   // countOnPump
    static uint16_t pcdp;   // countDayPump
    static uint8_t ptto;    // timeToOff
    static float pi;        // currentRMS
    static float pt;        // temperatures

    static uint8_t iNmsg;   // number error message to screen
    static uint8_t iTshow;  // counter times to show error message

    char* errMsgRCWPS[8] = {                    // mesages of errors from MCC
      " RelayON noVolt",
      " BlockON VoltON",
      "",
      "",
      "",
      "",
      "",
      ""
    };

    char errMsgMCC[8][16];  // 8 сообщений, каждое длиной до 16 символов
    // Преобразуем сообщения в массивы char
    snprintf(errMsgMCC[0], sizeof(errMsgMCC[0]), "Last ON > %dayMinute", maxOnPump / 60);
    snprintf(errMsgMCC[1], sizeof(errMsgMCC[1]), "Day ON > %dayMinute", maxDayPump / 60);
    snprintf(errMsgMCC[2], sizeof(errMsgMCC[2]), "I=%.2fA > %.2f", avgCurrent, maxCurrent);
    snprintf(errMsgMCC[3], sizeof(errMsgMCC[3]), "I=%.2fA < %.2f", avgCurrent, minCurrent);
    snprintf(errMsgMCC[4], sizeof(errMsgMCC[4]), "T=%.2fC > %.2f", temperature, maxTemp);
    strcpy(errMsgMCC[5], "");
    strcpy(errMsgMCC[6], "");
    strcpy(errMsgMCC[7], "");

  // redraw display only if parameters change
  if(par != alarmRCWPS || pam != alarmMCC || pnd != numScreen ||
    phour != hour || pminute != minute || pnop != numOnPump || ph != humidity ||
    pcop != countOnPump || pcdp != countDayPump ||
    ptto != timeToOff || pi != currentRMS || pt != temperature) {

    if(numScreen == 1) {
```

```
    // 1st string - Clock
    lcd.clear();                    //  Чистим дисплей
    lcd.setCursor(0, 0);            //  Устанавливаем курсор в позицию (0 столбец, 0 строка)
    lcdPrintZeros(String(hour), 0, 2, 0);   //  lcd.print(u == 1 ? "ON" : "OFF");
    lcd.setCursor(2, 0);
    lcd.print(":");
    lcdPrintZeros(String(minute), 3, 2, 0);
    // 1st string - Timer Day
    lcd.setCursor(6, 0);
    lcd.print("D");
    lcdPrintZeros(String(dayHour), 7, 2, 0);   // Hours from countOnDay
    lcd.setCursor(9, 0);
    lcd.print("h");
    lcdPrintZeros(String(dayMinute), 10, 2, 0);  // Minutes from countOnDay
    lcd.setCursor(12, 0);
    lcd.print("m");
    lcdPrintZeros(String(daySecond), 13, 2, 0);  // Seconds from countOnDay
    lcd.setCursor(15, 0);
    lcd.print("s");
    // 2nd string - Rest Time Pump is On
    lcd.setCursor(0, 1);
    lcd.print("R");
    lcdPrintZeros(String(timeToOff), 1, 3, 1);  // Rest time pump is on
    lcd.setCursor(4, 1);
    lcd.print("s");
    // 2nd string - number of starts
    lcd.setCursor(6, 1);
    lcd.print("N");
    lcd.setCursor(7, 1);
    lcd.print(numOnPump);
    // 2nd string - Time the last of pump start
    lcdPrintZeros(String(tMinute), 10, 2, 1);  // Time the last of pump start
    lcd.setCursor(12, 1);
    lcd.print("m");
    lcdPrintZeros(String(tSecond), 13, 2, 1);
    lcd.setCursor(15, 1);
    lcd.print("s");
}
if(numScreen == 2) {
    // 1st string - Clock
    lcd.clear();                    //  Чистим дисплей
    lcd.setCursor(0, 0);            //  Устанавливаем курсор в позицию (0 столбец, 0 строка)
    lcdPrintZeros(String(hour), 0, 2, 0);   //  lcd.print(u == 1 ? "ON" : "OFF");
    lcd.setCursor(2, 0);
    lcd.print(":");
    lcdPrintZeros(String(minute), 3, 2, 0);
    // 1st string - Temperature
    lcdPrintZeros(String(temperature), 6, 4, 0);  // Temperatures
    lcd.setCursor(10, 0);
    lcd.print("C");
    lcdPrintZeros(String(humidity), 13, 2, 0); // Humidity
    lcd.setCursor(15, 0);
    lcd.print("%");
    // 2nd string - RMS Current
    lcd.setCursor(0, 1);
    lcd.print("I");
    lcd.setCursor(1, 1);
    lcd.print(currentRMS);
    // 2nd string - number of starts
    lcd.setCursor(6, 1);
    lcd.print("N");
    lcd.setCursor(7, 1);
    lcd.print(numOnPump);        // Number of pump starts
    // 2nd string - Time the last of pump start
    lcdPrintZeros(String(tMinute), 10, 2, 1);  // Time the last of pump start
    lcd.setCursor(12, 1);
    lcd.print("m");
    lcdPrintZeros(String(tSecond), 13, 2, 1);
    lcd.setCursor(15, 1);
    lcd.print("s");
```

```
    }
  }
    if(numScreen == 3) {
       // output error message RCWPS on 1st string, MCC on 2nd string every 4 loop
       if(iTshow % 4 == 0){
         lcd.clear();                    //  Чистим дисплей
         lcd.setCursor(0, 0);
         lcd.print(iNmsg);
         if(codErrRCWPS[iNmsg] == true){lcd.print(errMsgRCWPS[iNmsg]);} else {lcd.print(" RCWPS is OK");}
         lcd.setCursor(0, 1);
         if(codErrMCC[iNmsg] == true){lcd.print(errMsgMCC[iNmsg]);} else {lcd.print(" MCC is OK");}
         iNmsg = (iNmsg + 1) % 5;           // counter number error message, to zero each 8
       }
       iTshow++;                        // counter show times error message
    }

  pv = voltageOn;
  par = alarmRCWPS;
  pam = alarmMCC;
  psop = stateOnPump;
  psb = stateBlock;
  pnd = numScreen;
  phour = hour;
  pminute = minute;
  pnop = numOnPump;
  ph = humidity;
  pcop = countOnPump;
  pcdp = countDayPump;
  ptto = timeToOff;
  pi = currentRMS;
  pt = temperature;

} // end toDisplay()

// Функция вывода текста с дополнением нулями
void lcdPrintZeros(String value, uint8_t colB, uint8_t width, uint8_t row) {
    String formattedValue = value;

    // Добавление нулей, если длина строки меньше требуемой ширины
    while (formattedValue.length() < width) {
        formattedValue = "0" + formattedValue;
    }

    // Ограничение длины строки (на случай ошибок)
    if (formattedValue.length() > width) {
        formattedValue = formattedValue.substring(0, width);
    }
    lcd.setCursor(colB, row);
    lcd.print(formattedValue);
} // end lcdPrintZeros()
```

Листинг 8 - Модуль дисплея

5.4.5 Таймер подсветки дисплея

Для отсчета времени подсветки дисплея использован аппаратный таймер 0 группы 0 микроконтроллера и стандартные библиотеки управления таймерами.

Сигнал срабатывания таймера по заданному значению:

- делитель частоты APB тактирования 40000, 80 МГц / 40000 = 2000 кГц;
- время одного такта 1 / 2000 кГц = 0,0005 сек;
- время работы таймера до срабатывания 240000 * 0,0005 = 120 сек.

В контроллере ESP32 уставки таймера позволяют организовать длительные выдержки времени. Поэтому он позволяет достичь выдержки 2 мин за один цикл.

Таймер и подсветка включаются по прикосновению к кнопке MENU, а отключаются уже в подпрограмме таймера после окончания отсчета.

```
// Выключение подсветки дисплея. функция прерывания от таймера (ISR). IRAM_ATTR sence is load function to RAM
void IRAM_ATTR onTimer(void* arg){
  timer_group_clr_intr_status_in_isr(TIMER_GROUP, TIMER_IDX); // Очистка флага прерывания таймера
  // код для обработки прерывания
  backLight = false;              //  Выключаем подсветку LCD дисплея
  flagLight = true;              // flag indicate that backLight is changed
  timer_pause(TIMER_GROUP, TIMER_IDX);  // Остановить таймер
  timer_set_alarm(TIMER_GROUP, TIMER_IDX, TIMER_ALARM_EN);  // Once triggered, the alarm is disabled automatically
and needs to be re-enabled to trigger again.
}

void setup(){
// ********************** Settings backlight timer LCD **********************
  timer_config_t timeconf = {
    .alarm_en = TIMER_ALARM_EN,
    .counter_en = TIMER_PAUSE,
    .intr_type = TIMER_INTR_LEVEL,
    .counter_dir = TIMER_COUNT_UP,
    .auto_reload = TIMER_AUTORELOAD_EN,
    .divider = TIMER_DIVIDER
  };
  timer_init(TIMER_GROUP, TIMER_IDX, &timeconf);
  timer_set_alarm_value(TIMER_GROUP, TIMER_IDX, TIMER_ALARM_VALUE); // Установка значения сигнализации
  timer_enable_intr(TIMER_GROUP, TIMER_IDX);  // Включить прерывания
  timer_isr_register(TIMER_GROUP, TIMER_IDX, onTimer, NULL, ESP_INTR_FLAG_IRAM, NULL); // Привязать обработчик
прерывания
}

void loop(){
//******************************** LCD ***************************************
  // Backlight LCD after MENU button pressing
    if(touchRead(buttonMenu) <= 25) {tMenu = true;} else {tMenu = false;}
    if(tMenu && !backLight && !flagLight){
    lcd.setBacklight(1);              //  Включаем подсветку LCD дисплея
    backLight = true;
    timer_start(TIMER_GROUP, TIMER_IDX);  // Запустить таймер
  }
  if (!tMenu && !backLight && flagLight) {
    lcd.setBacklight(0);              //  Выключаем подсветку LCD дисплея
    flagLight = false;
  }

}
```

Листинг 9 - Модуль таймера

5.4.6 Часы реального времени

Возможности системы контроля НАС значительно расширены применением модуля часов реального времени (RTC) с автономным питанием. Кроме отображения текущего времени он позволяет считать время наработки отдельно по дням, вести недельный учет и многое другое.

Кроме того, для счета времени наработки используется сигнал с отдельного вывода модуля частотой 1 Гц.

Точность отсчета часов зависит от точности установленного кварца. Установка времени реализована через веб-интерфейс, программная обработка приведена в модуле Веб сервер.

```
void setup(){
```

```
// ********************** Settings Real Time clock DS1307 ********************
  URTCLIB_WIRE.begin();
  rtc.set_rtc_address(0x68);
            rtc.set_model(rtcModel);  // set RTC Model
            rtc.refresh();  // refresh data from RTC HW in RTC class object so flags like rtc.lostPower(), rtc.getEOSCFlag(), etc, can
get populated
  // Only use once, then disable
  // RTCLib::set(byte second, byte minute, byte hour (0-23:24-hour mode only), byte dayOfWeek (Sun = 1, Sat = 7), byte
dayOfMonth (1-12), byte month, byte year)
  rtc.sqwgSetMode(URTCLIB_SQWG_1H); // Setting SQWG/INT output
  delay(10);
}

void loop(){
// refresh and read data from RTC HW in RTC class object so flags like rtc.lostPower(), rtc.getEOSCFlag(), etc, can get
populated
  rtc.refresh();
  year = rtc.year();
  month = rtc.month();
  day = rtc.day();
  hour = rtc.hour();
  minute = rtc.minute();
  dayHour = countDayPump/60/60;
  dayMinute = (countDayPump - dayHour * 60)/60;
  daySecond = countDayPump%60;
  tHour = countOnPump/60/60;
  tMinute = (countOnPump - tHour * 60)/60;
  tSecond = countOnPump%60;
}
```

Листинг 10 - Модуль часов реального времени

5.4.7 Счетчики времени наработки

Для счета времени наработки используется сигнал с отдельного выхода модуля RTC частотой 1 Гц. По этому сигналу вызывается аппаратное прерывание и выполняется подпрограмма обработчик события. Управление счетом осуществляется включением и отключением прерывания.

Особенностью подпрограмм аппаратных прерываний это необходимость быстрого их выполнения, а значит минимальное количество кода в них. Поэтому при прерывании устанавливается флаг и изменяется счетчик, а остальной алгоритм выполняется в основной программе.

Организованы следующие счетчики:

- счетчик времени последнего включения;
- счетчик времени наработки за текущие сутки;
- счетчик количества включений за текущие сутки.

Кроме часов модуль RTC оснащен флеш памятью AT24C32 (адресация ячеек от 1 до 56). Функции обращения к ней входят в библиотеку rtc. В модуле Часы реального времени выполняется чтение ранее сохраненных значений счетчиков. Запись значений счетчиков выполняется в этом модуле.

```
// function interrupt counters operating time from external wave 1Hz
void IRAM_ATTR counterOn(){
  countOnPump++;
  flagPulse = true;
}
```

```
void setup(){
// ************************** Read stored data *********************************
  currentDay = rtc.ramRead(ADRDAY);     // Read currentDay in I2C AT24C32 memory (addr: 1 - 56)
  numOnPump = rtc.ramRead(ADRNUMON);     // Read numOnPump in I2C AT24C32 memory (addr: 1 - 56)
  countOnPump = (rtc.ramRead(ADRCOUNT+1) << 8) | rtc.ramRead(ADRCOUNT);  // Read countOnPump in I2C AT24C32
memory (addr: 1 - 56)
  countDayPump = (rtc.ramRead(ADRCOUNTDAY+1) << 8) | rtc.ramRead(ADRCOUNTDAY);    // Read countDayPump in I2C
AT24C32 memory (addr: 1 - 56)
}

void loop(){
// ************* Counters working time ****************************
  // ON interrupt from 1Hz signal
  if(voltageOn && !prevStateOnPump && !flagCount){
    attachInterrupt(digitalPinToInterrupt(rtcSQPin), counterOn, FALLING);
    flagCount = true;
    countOnPump = 0;
    numOnPump++;
    prevStateOnPump = true;
    rtc.ramWrite(ADRNUMON, numOnPump);   // Write numOnPump in I2C AT24C32 memory (addr: 1 - 56)
  }
  // OFF interrupt from 1Hz signal
  if(!voltageOn && !stateOnPump && prevStateOnPump && flagCount){
    detachInterrupt(digitalPinToInterrupt(rtcSQPin));
    flagCount = false;
    prevStateOnPump = false;
    lowByte = countOnPump & 0xFF;          // Младший байт
    highByte = (countOnPump >> 8) & 0xFF;   // Старший байт
    rtc.ramWrite(ADRCOUNT, lowByte);        // Write countOnPump in I2C AT24C32 memory (addr: 1 - 56)
    rtc.ramWrite(ADRCOUNT+1, highByte);     // Write countOnPump in I2C AT24C32 memory (addr: 1 - 56)
    lowByte = countDayPump & 0xFF;          // Младший байт
    highByte = (countDayPump >> 8) & 0xFF;  // Старший байт
    rtc.ramWrite(ADRCOUNTDAY, lowByte);         // Write countDayPump in I2C AT24C32 memory (addr: 1 - 56)
    rtc.ramWrite(ADRCOUNTDAY+1, highByte);       // Write countDayPump in I2C AT24C32 memory (addr: 1 - 56)
  }
  // continue interrupt handling 1Hz counter
  if(flagPulse){
    if(currentDay == rtc.day()){
      countDayPump++;
    } else {
      countDayPump = 0;
      numOnPump = 0;
      currentDay = rtc.day();
      rtc.ramWrite(ADRDAY, currentDay);   // Reset numOnPump in I2C AT24C32 memory (addr: 1 - 56)
      rtc.ramWrite(ADRNUMON, 0);   // Reset numOnPump in I2C AT24C32 memory (addr: 1 - 56)
      rtc.ramWrite(ADRCOUNTDAY, 0);    // Reset countDayPump in I2C AT24C32 memory (addr: 1 - 56)
      rtc.ramWrite(ADRCOUNTDAY+1, 0);    // Reset countDayPump in I2C AT24C32 memory (addr: 1 - 56)
    }
    flagPulse = false;
  }
}
```

Листинг 11 - Модуль счетчиков

5.4.8 Управление и индикация

Управляющая часть программы определяет команду от кнопок или от веб-интерфейса и устанавливает соответствующую переменную для дальнейшей передачи по радиоканалу на периферийный контроллер для исполнения.

Команда блокировки работает в режиме триггера. Для этого контроллера организован программный триггер.

```
void loop(){
//********************** Control part - LED *********************************
  // Control LED pump state. Green
```

```
  if(voltageOn){
    digitalWrite(ledOnPumpPin, HIGH);
  } else {
    digitalWrite(ledOnPumpPin, LOW);
  }
  // Control LED pump block state. Red
  if(stateBlock){
    digitalWrite(ledBlockPin, HIGH);
  } else {
    digitalWrite(ledBlockPin, LOW);
  }
  // Control LED alarm state. Yellow
  if(stateAlarm){
    digitalWrite(ledAlarmPin, HIGH);
  } else {
    digitalWrite(ledAlarmPin, LOW);
  }

//******************* Control part - Buttons  ***************************
  // command ON Pump
  pumpON = !digitalRead(buttonOnPumpPin) || webOn;
  // command OFF Pump
  pumpOFF = !digitalRead(buttonOffPumpPin) || webOff;

  // Triger that switch by pressing BLOCK button
  if((!digitalRead(buttonBlockPin) == true && !digitalRead(buttonBlockPin) != prevBlock && trigerBlock == false) ||
    (webBlock == true && webBlock != prevBlock && trigerBlock == false)) {
    trigerBlock = true;
  } else if((!digitalRead(buttonBlockPin) == true && !digitalRead(buttonBlockPin) != prevBlock && trigerBlock == true) ||
    (webBlock == true && webBlock != prevBlock && trigerBlock == true)){
    trigerBlock = false;
  }
  prevBlock = !digitalRead(buttonBlockPin) || webBlock;
}
```

Листинг 12 - Модуль управления и индикации

5.4.9 Обработка аварийных ситуаций

Функции обработки аварийных ситуаций поделены между двумя контроллерами. Контроллер МСС учитывает ошибки для определения которых требуется использовать изменяемые или вычисляемые параметры.

Перечень ошибок указан в главе Пакетная передача информации.

```
void loop(){
//********************** Alarms ***********************************
  stateAlarm = alarmMCC || alarmRCWPS;
  // Check max time working pump
  if(countOnPump > maxOnPump){
    codErrMCC[0] = true;
  } else {
    codErrMCC[0] = false;
  }
  // Check max day time working pump
  if(countDayPump > maxDayPump){
    codErrMCC[1] = true;
  } else {
    codErrMCC[1] = false;
  }

  // Calculate average current
  avgCurrent = calcAverage(currentRMS);
  // Check max working rms current
  if(avgCurrent > maxCurrent){
    codErrMCC[2] = true;
```

```
  } else {
    codErrMCC[2] = false;
  }
  // Check min working rms curren t
  if(voltageOn && avgCurrent < minCurrent){
    codErrMCC[3] = true;
  } else {
    codErrMCC[3] = false;
  }
  // Check max working temperatures
  if(temperature > maxTemp){
    codErrMCC[4] = true;
  } else {
    codErrMCC[4] = false;
  }

  // Set flag alarmMCC if in array codErrMCC error
  alarmMCC = false;
  for (uint8_t i = 0; i < 8; i++) {
    alarmMCC = alarmMCC || codErrMCC[i];  // Выполняем логическое ИЛИ
    if (alarmMCC) break;                  // Если результат уже true, можно прервать цикл
  }
}

// function calculate average value from several (NUM_AVG)
float calcAverage(float current) {
  static float values[NUM_AVG] = {0};       // Буфер для хранения последних значений
  static int index = 0;                  // Текущий индекс в буфере
  static int count = 0;                  // Количество заполненных элементов (до 10)
  //int current = round(10 * val);          // convert to int

  // Записываем новое значение в буфер
  values[index] = current;
  index = (index + 1) % NUM_AVG; // Перемещаем индекс по кольцу

  // Увеличиваем счетчик заполненных элементов, но не больше NUM_VALUES
  if (count < NUM_AVG) {
    count++;
  }

  // Вычисляем среднее значение
  float sum = 0;
  for (int i = 0; i < count; i++) {
    sum += values[i];
  }
  // static_cast<float>(sum) / count / 10;  // convert to float
  return sum / count;
} // end calcAverage()
```

Листинг 13 - Модуль определения аварийных ситуаций

5.4.10 SD-MMC память

В проекте используется SD-MMC карта памяти для хранения страниц веб-интерфейса. К модулю SD-MMC можно отнести раздел инициализации, а чтение из карты памяти выполняется в модуле Веб сервер.

В дальнейшем развитии проекта предполагается тут же хранить массив данных за продолжительный отрезок времени для построения графиков и анализа.

```
void setup(){
// ****************************** Settings SD-MMC card ************************************
  if (!SD_MMC.begin("/sdcard", true)) {
    Serial.println("Card Mount Failed");
    return;
  }
```

```
  uint8_t cardType = SD_MMC.cardType();
  if (cardType == CARD_NONE) {
    Serial.println("No SD_MMC card attached");    // !!!!!!!!!!!!!!!
    return;
  }
  Serial.print("SD_MMC Card Type: ");
  if (cardType == CARD_MMC) {
    Serial.println("MMC");
  } else if (cardType == CARD_SD) {
    Serial.println("SDSC");
  } else if (cardType == CARD_SDayHourC) {
    Serial.println("SDayHourC");
  } else {
    Serial.println("UNKNOWN");
  }
  Serial.printf("Total space: %lluMB\n", SD_MMC.totalBytes() / (1024 * 1024));
  Serial.printf("Used space: %lluMB\n", SD_MMC.usedBytes() / (1024 * 1024));
  // check for index.html file
  if (!SD_MMC.exists("/index.html")) {
      Serial.println("ERROR - Can't find index.html file!");
      return;  // can't find index file
  }
  delay(10);
}
```

Листинг 14 - Модуль SD-MMC

5.4.11 Wi-Fi

В проекте используется Wi-Fi подключение к локальной сети для возможности доступа к странице веб-интерфейса системы контроля НАС. К модулю Wi-Fi можно отнести раздел инициализации, а обращение к его объектам выполняется в модуле Веб сервер.

```
void setup(){
// ************************** Settings Wi-Fi ************************************
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);             //  Network.begin(mac, ip);  // initialize Network device
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected.");

  // Print connections parameters
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.print("Hostname: ");
  Serial.println(WiFi.getHostname());
  Serial.print("ESP Mac Address: ");
  Serial.println(WiFi.macAddress());
  Serial.print("Subnet Mask: ");
  Serial.println(WiFi.subnetMinuteask());
  Serial.print("Gateway IP: ");
  Serial.println(WiFi.gatewayIP());
  Serial.print("DNS: ");
  Serial.println(WiFi.dnsIP());

  server.begin();                     // start to listen for clients
}
```

Листинг 15 - Модуль Wi-Fi

5.4.12 Веб сервер

В проекте значительная часть программы занимает организация веб сервера и обработка информационного обмена.

При подключении клиента ему передается index.html файл, считанный из карты памяти. В нем содержится css оформление и программа обработки запроса параметров по стандарту XMLHttpRequest на javascript.

Подпрограма webCommand обрабатывает команды START, STOP, BLOCK_WORK, Se_Time поступившие от клиента в запросе GET.

Подпрограма XML_response подготавливает и передает данные клиенту.

Подпрограмма setTimeFromHttp устанавливает время часов реального времени по времени на часах клиента.

```
void setup(){
NetworkClient client = server.accept();       // try to get client

  if (client) {                               // got client?
     boolean currentLineIsBlank = true;
     while (client.connected()) {
        if (client.available()) {             // client data available to read
           char c = client.read();            // read 1 byte (character) from client
           // limit the size of the stored received HTTP request
           // buffer first part of HTTP request in HTTP_req array (string)
           // leave last element in array as 0 to null terminate string (REQ_BUF_SZ - 1)
           if (req_index < (REQ_BUF_SZ - 1)) {
              HTTP_req[req_index] = c;       // save HTTP request character
              req_index++;
           }
           // last line of client request is blank and ends with \n
           // respond to client only after last line received
           if (c == '\n' && currentLineIsBlank) {
              // send a standard http response header
              client.println("HTTP/1.1 200 OK");
              // remainder of header follows below, depending on if
              // web page or XML page is requested
              // Ajax request - send XML file
              if (StrContains(HTTP_req, "ajax_inputs")) {
                 // send rest of HTTP header
                 client.println("Content-Type: text/xml");
                 client.println("Connection: keep-alive");
                 client.println();
                 webCommand();
                 XML_response(client);                    // send XML file containing input states
              } else {                                    // web page request
                 // send rest of HTTP header
                 client.println("Content-Type: text/html");
                 client.println("Connection: keep-alive");
                 client.println();
                 // send web page
                 webFile = SD_MMC.open("/index.html");           // open web page file
                 if (webFile) {
                    while(webFile.available()) {
                       client.write(webFile.read());       // send web page to client
                    }
                    webFile.close();
                 }
              }
              // display received HTTP request on serial port
              Serial.print(HTTP_req);
              // reset buffer index and all buffer elements to 0
              req_index = 0;
              StrClear(HTTP_req, REQ_BUF_SZ);
              break;
```

```
            }
            // every line of text received from the client ends with \r\n
            if (c == '\n') {
               // last character on line of received text
               // starting new line with next character read
               currentLineIsBlank = true;
            } else if (c != '\r') {
               // a text character was received from client
               currentLineIsBlank = false;
            }
         } // end if (client.available())
      } // end while (client.connected())
      delay(1);      // give the web browser time to receive the data
      client.stop(); // close the connection
  } // end if (client)
}

// Resolve commands from web client
void webCommand(void){
  // Start
    if (StrContains(HTTP_req, "START")) {
      webOn = true;
    } else {
      webOn = false;
    }
  // Stop
    if (StrContains(HTTP_req, "STOP")) {
      webOff = true;
    } else {
      webOff = false;
    }
  // Block
    if (StrContains(HTTP_req, "BLOCK_WORK")) {
      webBlock = true;
    } else {
      webBlock = false;
    }
  // Set Time
    if (StrContains(HTTP_req, "second=")) {
      setTimeFromHttp(HTTP_req);
    }
} // end webCommand()

// send the XML file with analog values, switch status
void XML_response(NetworkClient cl){
    cl.println("<?xml version=\"1.0\" ?>");
    cl.println("<root>");

    cl.print("<data>");
    cl.print(addZero(String(day), 2) + "-" + addZero(String(month), 2) + "-20" + addZero(String(year), 2));
    cl.println("</data>");

    cl.print("<time>");
    cl.print(addZero(String(hour), 2) + ":" + addZero(String(minute), 2));
    cl.println("</time>");

    cl.print("<temperature>");
    cl.print(String(temperature));
    cl.println("</temperature>");

    cl.print("<humidity>");
    cl.print(String(humidity));
    cl.println("</humidity>");

    cl.print("<voltage>");
    cl.print(voltageOn == true ? "ON" : "OFF");
    cl.println("</voltage>");

    cl.print("<rmsCurrent>");
    cl.print(String(currentRMS));
```

```
  cl.println("</rmsCurrent>");

  cl.print("<pressure>");
  cl.print(String(pressure));
  cl.println("</pressure>");

  cl.print("<lastOnTime>");
  cl.print(addZero(String(tHour), 2) + "h " + addZero(String(tMinute), 2) + "m " + addZero(String(tSecond), 2) + "s");
  cl.println("</lastOnTime>");

  cl.print("<timePerDay>");
  cl.print(addZero(String(dayHour), 2) + "h " + addZero(String(dayMinute), 2) + "m " + addZero(String(daySecond), 2) + "s");
  cl.println("</timePerDay>");

  cl.print("<startsPerDay>");
  cl.print(String(numOnPump));
  cl.println("</startsPerDay>");

  cl.print("<timeToOff>");
  cl.print(addZero(String(timeToOff), 3) + "s");
  cl.println("</timeToOff>");

  cl.print("<block>");
  cl.print(String(stateBlock == true ? "ON" : "OFF"));
  cl.println("</block>");

  cl.print("<alarm>");
  cl.print(String(stateAlarm == true ? "ON" : "OFF"));
  cl.println("</alarm>");

  cl.print("<relay>");
  cl.print(String(stateOnPump == true ? "ON" : "OFF"));
  cl.println("</relay>");

  cl.println("</root>");
} // end XML_response()

// Function set time to DS1307 from client http request string
void setTimeFromHttp(String http_req) {
  uint8_t second;
  uint8_t minute;
  uint8_t hour;
  uint8_t dayOfWeek;
  uint8_t dayOfMonth;
  uint8_t month;
  uint8_t year;

  int paramStart = http_req.indexOf('?');
  if (paramStart == -1) {
    Serial.println("Ошибка: строка запроса не найдена!");
    return;
  }

  // Извлекаем параметры
  String params = http_req.substring(paramStart + 1);
  int paramEnd = params.indexOf(' ');
  if (paramEnd != -1) {
    params = params.substring(0, paramEnd);
  }

  // Разбиваем параметры по '&'
  while (params.length() > 0) {
    int eqIndex = params.indexOf('=');
    int ampIndex = params.indexOf('&');

    if (eqIndex != -1) {
      String key = params.substring(0, eqIndex);
      String value = (ampIndex != -1) ? params.substring(eqIndex + 1, ampIndex) : params.substring(eqIndex + 1);

      if (key == "second") second = value.toInt();
```

```
      else if (key == "minute") minute = value.toInt();
      else if (key == "hour") hour = value.toInt();
      else if (key == "dayOfWeek") dayOfWeek = value.toInt();
      else if (key == "dayOfMonth") dayOfMonth = value.toInt();
      else if (key == "month") month = value.toInt();
      else if (key == "year") year = value.toInt();

      if (ampIndex == -1) break; // Если больше нет параметров, выходим из цикла
      params = params.substring(ampIndex + 1); // Убираем обработанный параметр
    } else {
      break;
    }
  }

  // Устанавливаем время на модуле DS1307
  // RTCLib::set(byte second, byte minute, byte hour (0-23:24-hour mode only), byte dayOfWeek (Sun = 1, Sat = 7), byte
dayOfMonth (1-12), byte month, byte year)
  rtc.set(second, minute, hour, dayOfWeek, dayOfMonth, month, year);
  Serial.println("Время успешно установлено!");
  Serial.println();
} // end setTimeFromHttp()

// Функция дополнения текста нулями
String addZero(String value, uint8_t width) {
    String formattedValue = value;

    // Добавление нулей, если длина строки меньше требуемой ширины
    while (formattedValue.length() < width) {
        formattedValue = "0" + formattedValue;
    }

    // Ограничение длины строки (на случай ошибок)
    if (formattedValue.length() > width) {
        formattedValue = formattedValue.substring(0, width);
    }
    return formattedValue;
} // end addZero()

// sets every element of str to 0 (clears array)
void StrClear(char *str, char length)
{
    for (int i = 0; i < length; i++) {
        str[i] = 0;
    }
}

// searches for the string sfind in the string str
// returns 1 if string found
// returns 0 if string not found
char StrContains(const char *str, const char *sfind)
{
    char found = 0;
    char index = 0;
    char len;

    len = strlen(str);

    if (strlen(sfind) > len) {
        return 0;
    }
    while (index < len) {
        if (str[index] == sfind[found]) {
            found++;
            if (strlen(sfind) == found) {
                return 1;
            }
        }
        else {
            found = 0;
        }
```

```
        index++;
    }
    return 0;
} // end StrContains()
```

Листинг 16 - Модуль Веб сервера

5.4.13 Веб клиент

После подключения к серверу клиент получает html страницу https://github.com/Ochilnik/WPS/blob/master/index.html со скриптом AJAX запроса для получения данных с сервера на JavaScript.

Назначение этого скрипта — онлайн обновление отдельных параметров, изменение оформления и т.п. на уже открытой странице без полной её перерисовки. Скрипт получает обновляемые раз в 1 сек параметры системы и передает предусмотренные команды. Все общение осуществляется через веб-интерфейс в браузере клиента.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Water pumping and storage station</title>

    <script>
        var data = "";
        var time = "";
        var temperature = "";
        var humidity = "";

        var voltage = "";
        var rmsCurrent = "";
        var pressure = "";

        var lastOnTime = "";
        var timePerDay = "";
        var startsPerDay = "";
        var timeToOff = "";

        var block = "";
        var alarm = "";
        var relay = "";

        var strCommand = "";

        // AJAX запрос для получения данных с сервера
        function getDataFromServer() {
            var nocache = "&nocache=" + Math.random() * 1000000;
            var request = new XMLHttpRequest();
            request.onreadystatechange = function() {
                if (this.readyState == 4) {
                    if (this.status == 200) {
                        if (this.responseXML != null) {
                            // Получаем данные с сервера
                            data = this.responseXML.getElementsByTagName('data')[0].childNodes[0].nodeValue;
                            time = this.responseXML.getElementsByTagName('time')[0].childNodes[0].nodeValue;
                            temperature = this.responseXML.getElementsByTagName('temperature')[0].childNodes[0].nodeValue;
                            humidity = this.responseXML.getElementsByTagName('humidity')[0].childNodes[0].nodeValue;

                            voltage = this.responseXML.getElementsByTagName('voltage')[0].childNodes[0].nodeValue;
                            rmsCurrent = this.responseXML.getElementsByTagName('rmsCurrent')[0].childNodes[0].nodeValue;
                            pressure = this.responseXML.getElementsByTagName('pressure')[0].childNodes[0].nodeValue;
```

```
                    lastOnTime = this.responseXML.getElementsByTagName('lastOnTime')[0].childNodes[0].nodeValue;
                    timePerDay = this.responseXML.getElementsByTagName('timePerDay')[0].childNodes[0].nodeValue;
                    startsPerDay = this.responseXML.getElementsByTagName('startsPerDay')[0].childNodes[0].nodeValue;
                    timeToOff = this.responseXML.getElementsByTagName('timeToOff')[0].childNodes[0].nodeValue;

                    block = this.responseXML.getElementsByTagName('block')[0].childNodes[0].nodeValue;
                    alarm = this.responseXML.getElementsByTagName('alarm')[0].childNodes[0].nodeValue;
                    relay = this.responseXML.getElementsByTagName('relay')[0].childNodes[0].nodeValue;

                    // Обновляем элементы на странице
                    document.getElementById("data").textContent = "Data: " + data;
                    document.getElementById("time").textContent = "Time: " + time;
                    document.getElementById("temperature").textContent = "Temperature: " + temperature + "°C";
                    document.getElementById("humidity").textContent = "Humidity: " + humidity + "%";

                    document.getElementById("voltage").value = voltage;
                    document.getElementById("voltage").style.backgroundColor = voltage === "ON" ? "#00FF00" : "#FFFFFF"; //
Зелёный, если ON, иначе белый
                    document.getElementById("rms").value = rmsCurrent;
                    document.getElementById("pressure").value = pressure;

                    document.getElementById("last-time").value = lastOnTime;
                    document.getElementById("time-per-day").value = timePerDay;
                    document.getElementById("starts-per-day").value = startsPerDay;
                    document.getElementById("time-to-off").textContent = timeToOff;

                    // Обновляем цвета кнопок
                    document.getElementById("start-btn").style.backgroundColor = relay === "ON" ? "#00FF00" : "#C2DFFF";
                    document.getElementById("stop-btn").style.backgroundColor = alarm === "ON" ? "yellow" : "#C2DFFF";
                    document.getElementById("block-btn").style.backgroundColor = block === "ON" ? "red" : "#C2DFFF";
                }
            }
        }
    };
    // Отправляем запрос на сервер
    request.open("GET", "ajax_inputs?" + strCommand + nocache, true);
    request.send(null);
    setTimeout(getDataFromServer, 1000); // Повторяем запрос каждую секунду
}

// PRESS BUTTONS
function onStartClick() {
    strCommand = "action=START";
    getDataFromServer();
    strCommand = "";
}
function onStopClick() {
    strCommand = "action=STOP";
    getDataFromServer();
    strCommand = "";
}
function onBlockClick() {
    strCommand = "action=BLOCK_WORK";
    getDataFromServer();
    strCommand = "";
}
function setTime() {
    const now = new Date();
    // Получаем параметры времени
    const second = now.getSeconds();
    const minute = now.getMinutes();
    const hour = now.getHours(); // В формате 0-23
    const dayOfWeek = now.getDay() === 0 ? 7 : now.getDay(); // Преобразуем из 0-6 (в JS) в 1-7
    const dayOfMonth = now.getDate();
    const month = now.getMonth() + 1; // В JS месяцы начинаются с 0
    const year = now.getFullYear() % 100; // Берем последние две цифры года
    // Формируем строку запроса
    strCommand = `second=${second}&minute=${minute}&hour=${hour}&dayOfWeek=${dayOfWeek}&dayOfMonth=$
{dayOfMonth}&month=${month}&year=${year}`;
    getDataFromServer();
```

```
        strCommand = "";
    }


    // Запускаем получение данных при загрузке страницы
    window.onload = function() {
        getDataFromServer();
    };
</script>

<style>
    body {
        font-family: Arial, sans-serif;
        text-align: center;
        margin: 0;
        padding: 0;
        background-color: #C2DFFF; /* Синее море фон страницы */
    }

    h1 {
        margin-top: 50px;
    }

    .variables {
        display: flex;
        justify-content: space-evenly;
        align-items: center;
        margin-top: 20px;
        flex-wrap: wrap;
    }

    .variable {
        font-size: 18px;
        margin: 0 20px;
    }

    .boxes-container {
        display: flex;
        justify-content: space-evenly;
        margin-top: 40px;
        flex-wrap: wrap;
    }

    .box {
        border: 2px solid black;
        padding: 20px;
        width: 30%;
        box-sizing: border-box;
        margin: 10px;
    }

    .box h2 {
        margin-top: 0;
    }

    .param {
        margin: 10px 0;
    }

    .param label {
        margin-right: 10px;
        font-size: 18px;
    }

    .box button {
        margin-top: 10px;
        padding: 10px;
        width: 100%;
        font-size: 18px;
        font-weight: bold;
```

```
        }

        .input-field {
            padding: 5px;
            width: 90%;
            font-size: 18px;
            font-weight: bold;
            text-align: center; /* Значения в полях по центру */
            background-color: #FFFFFF; /* Начальный фон — белый */
        }

        .box:nth-child(1) {
            background-color: #3BB9FF; /* Глубокий небесно-голубой фон для PUMP */
        }

        .box:nth-child(2) {
            background-color: #FFDB58; /* горчичный фон для TIMERS */
        }

        .box:nth-child(3) {
            background-color: #FEFCFF; /* Молочный белый фон для FORCED */
        }

        /* Медиазапрос для вертикального формата */
        @media (max-width: 768px) {
            .boxes-container {
                flex-direction: column; /* Вертикальное расположение боксов */
                align-items: center;
            }

            .box {
                width: 80%; /* Уменьшаем ширину боксов на мобильных устройствах */
            }
        }

        #time-to-off {
            margin-left: 10px;
            font-size: 18px;
            font-weight: bold;
        }

        .set-time-btn {
            padding: 8px 16px;
            font-size: 16px;
            margin-left: 10px;
        }
        .right-par {
            text-align: right;
            font-family: verdana;
            font-size: 15px;
            margin: 10px;
            margin-top: 50px;
        }
    </style>

</head>

<body>

    <h1>Water pumping and storage station</h1>

    <div class="variables">
        <div class="variable" id="temperature">Temperature: ...</div>
        <div class="variable" id="humidity">Humidity: ...</div>
        <div class="variable" id="data">Data: ...</div>
        <div class="variable" id="time">Time: ...</div>
        <button class="set-time-btn" onclick="setTime()">Set Time</button>
    </div>

    <div class="boxes-container">
```

```
    <!-- PUMP Box -->
    <div class="box">
      <h2>PUMP</h2>
      <div class="param">
        <label for="voltage">Voltage is:</label>
        <input type="text" id="voltage" class="input-field">
      </div>
      <div class="param">
        <label for="rms">RMS current:</label>
        <input type="text" id="rms" class="input-field">
      </div>
      <div class="param">
        <label for="pressure">Pressure:</label>
        <input type="text" id="pressure" class="input-field">
      </div>
    </div>

    <!-- TIMERS Box -->
    <div class="box">
      <h2>TIMERS</h2>
      <div class="param">
        <label for="last-time">Last on time:</label>
        <input type="text" id="last-time" class="input-field">
      </div>
      <div class="param">
        <label for="time-per-day">Time per Day:</label>
        <input type="text" id="time-per-day" class="input-field">
      </div>
      <div class="param">
        <label for="starts-per-day">Number starts per day:</label>
        <input type="text" id="starts-per-day" class="input-field">
      </div>
    </div>

    <!-- FORCED Box -->
    <div class="box">
      <h2>FORCED</h2>
      <button id="start-btn" onclick="onStartClick()">START <span id="time-to-off"></span></button>
      <button id="stop-btn" onclick="onStopClick()">STOP</button>
      <button id="block-btn" onclick="onBlockClick()">BLOCK WORK</button>
    </div>
  </div>

  <footer>
                <p class="right-par">&copy; Ochilnik - WPS 2025</p>
        </footer>

</body>
</html>
```

Листинг 17 - Модуль Веб интерфейса

6 Описание работы

Основное время работы системы это контроль состояния НАС, подсчет времени наработки, отображение параметров и сигнализация аварийных состояний.

Подсчет времени работы ведется по наличию питающего напряжения на НАС. После отключения НАС счетчики хранят последние значения. Организованы следующие счетчики:

Счетчик времени последнего включения учитывает время работы насоса за одно включение. Сбрасывается каждый раз при начале нового счета.

Счетчик времени наработки за текущие сутки учитывает суточное время работы насоса. Сбрасывается после первого включения следующего дня.

Счетчик количества включений за текущие сутки. Сбрасывается после первого включения следующего дня.

Обратный счетчик времени оставшегося до выключения насоса после его пуска командой START. После окончания выдержки (1 мин) или после команды STOP сбрасывается в исходное значение 60 сек.

Текущие значения счетчиков отображаются на экране 1 дисплея МСС контроллера и на веб станице.

Рисунок 22 – Назначение индикации и управления МСС

Из остальных параметров контролируется среднеквадратичный ток потребления электродвигателя НАС, температура и влажность воздуха в помещении НАС. Эти параметры отображаются на экране 2 дисплея МСС контроллера и на веб станице.

Рисунок 23 – Назначение индикации экрана 2 дисплея МСС

Экран 3 дисплея МСС контроллера отображает по кругу сообщения об ошибках.

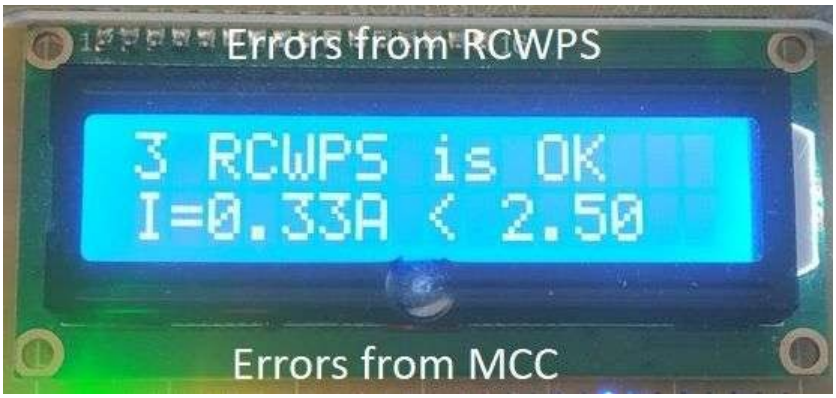

Рисунок 24 – Назначение индикации экрана 3 дисплея МСС

На текущую реализацию предусмотрено определение и сигнализация следующих ошибок:

Таблица 5 – Перечень ошибок индицируемых системой контроля НАС

№ п/п	Наименование ошибки	Сообщение на дисплее
1	После включения насоса командой START отсутствует напряжение питания	RelayON noVolt
2	После включения блокировки включения командой BLOCK WORK присутствует напряжение питания	BlockON VoltON
3	Время разового включения превышает 10 мин	Last ON > 10 min
4	Время работы за текущий день превышает 1 час	Day ON > 60 min
5	Средний потребляемый электродвигателем ток превышает 2,95А (перегрузка электродвигателя, Iном=2,8А)	I=X.XXA > 2,95A
6	Средний потребляемый электродвигателем ток ниже 2,5А (по данному значению определяется работа насоса вхолостую)	I=X.XXA < 2,5A
7	Температура в помещении выше 30С	T=XX.XC > 30C

Кроме дисплея на панели контроллера МСС предусмотрено три светодиодных индикатора.

Voltage – индицирует поданное напряжение (зеленый).

Error – индицирует состояние ошибки (желтый).

Block – индицирует включенную блокировку работы насоса (красный).

Контроллер RCWPS имеет индикатор состояния ошибки, индикатор напряжения AC220B, индикаторы Start ON и Block ON. Кнопки контроллера оснащены собственной индикацией.

Рисунок 25 – Назначение индикации и управления RCWPS

Предусмотрено форсированное управления НАС для принудительного включения НАС и принудительной блокировки его включения независимо от состояния управляющего насосом реле давления.

Органы управления предусмотрены на обоих контроллерах и повторяются в веб интерфейсе. Назначение кнопок:

START – пуск насоса на ограниченное время (1 мин). Ограничение сделано с целью предотвратить аварийные ситуации при длительной работе насоса, поскольку включение происходит в обход реле давления;

STOP – отключение насоса включенного предыдущей командой. Не влияет на работу, если насос включен от реле давления;

BLOCK WORK – блокировка включения насоса. Если насос включен останавливает его. Кнопка работает в триггерном режиме, для включения необходимо повторное нажатие.

MENU – есть только на центральном контроллере. Предназначена для переключения экранов на дисплее. Так же включает подсветку дисплея на ограниченное время (2 мин). Для облегчения доступа выполнена как сенсорная кнопка, электродом прикосновения выступает правый нижний винт крепления.

Веб интерфейс повторяет вышеуказанные органы управления и индикации. Индикация и управление разбиты на группы: Дата и время, Параметры НАС, Счетчики, Управление.

Через веб-интерфейс кнопкой Set Time возможно синхронизировать время внутренних часов системы контроля с временем клиента.

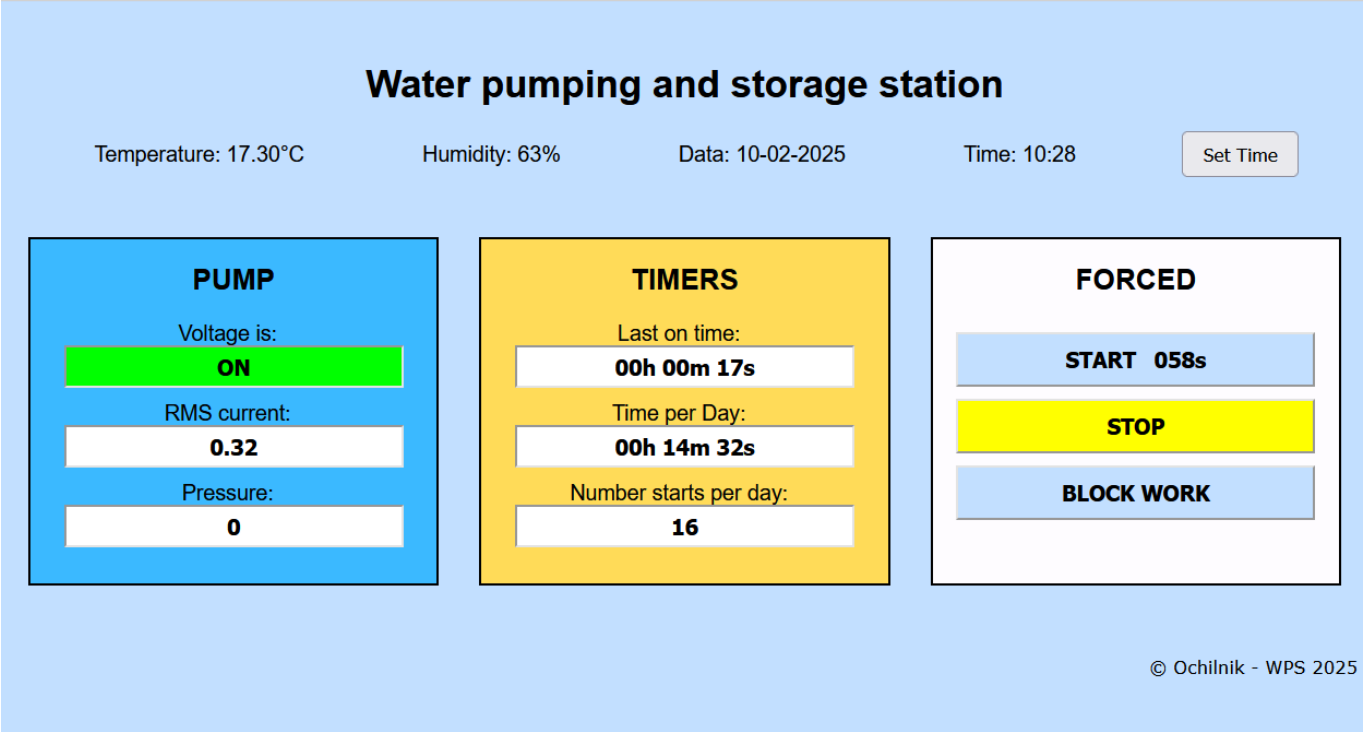

Рисунок 26 – Веб-интерфейс системы контроля НАС

Индикация состояний выполнена подсветкой определенных полей.

Таблица 6 – Отображение индикации состояний системы на веб-интерфейсе

№ п/п	Индикация	Значение
1	Зеленый цвет поля Voltage is	Напряжение подано
2	Желтый цвет поля RMS Current	Отклонение тока
3	Желтый цвет поля Last on time	Время превысило 10 мин
4	Желтый цвет поля Time per Day	Время превысило 60 мин
5	Зеленый цвет кнопки START	Форсированное включение
6	Желтый цвет кнопки STOP	Ошибка
7	Красный цвет кнопки BLOCK	Блокировка включена

Совместная работа всех компонентов системы на стенде продемонстрирована в следующем видео https://github.com/Ochilnik/WPS/blob/master/WPS.mkv.

7 Перспективы модернизации

Описанные возможности системы контроля НАС не являются совершенными и окончательными. Можно сказать, это первая версия, готовая для эксплуатационных испытаний.

Не все заложенные замыслы реализованы. Некоторые идеи появились в процессе реализации, некоторые в процессе написания этого отчета, некоторые еще появятся. Из основных перспектив модернизации хотелось бы отметить:

Оснащение НАС датчиком давления с аналоговым выходом. Информация о давлении позволит:

- однозначно реагировать на изменение давления за рабочими пределами;
- учитывать смещение рабочих пределов в зависимости от температуры окружающего воздуха;
- заранее определить падение давления воздуха в аккумуляторе;
- полностью реализовать функции интеллектуального реле давления.

Возможно измерения давления воды гидравлическим датчиком, возможно измерение давления воздуха пневматическим датчиком давления в аккумуляторе.

Для этого зарезервирован аналоговый вход на контроллере RCWPS, место в протоколе обмена, поле для индикации и визуализации.

Оснащение НАС дискретным датчиком протечки. Позволит быстро и однозначно реагировать на аварийные ситуации отключением насоса.

Оптимизация сетевого обмена с клиентом. Позволит разгрузить не сильно производительный сетевой интерфейс контроллера МСС для дополнительных задач.

Создание и ведение базы данных параметров НАС. Позволит хранить данные о работе за длительный период времени, получать из этих данных выборку, строить графики, сравнительные зависимости и др.